# Algorithmic Aspects of Machine Learning

Ankur Moitra

# Contents

# Preface

The monograph is based on the class "18.S996: Algorithmic Aspects of Machine Learning" taught at MIT in Fall 2013. Thanks to the scribes Adam Hesterberg, Adrian Vladu, Matt Coudron, Jan-Christian Hütter, Henry Yuen, Yufei Zhao, Hilary Finucane, Matthew Johnson, Kayhan Batmanghelich, Gautam Kamath, George Chen, Pratiksha Thaker, Mohammad Bavarian, Vlad Firoiu, Madalina Persu, Cameron Musco, Christopher Musco, Jing Lin, Timothy Chu, Yin-Tat Lee, Josh Alman, Nathan Pinsker and Adam Bouland.

# Chapter 1

# Introduction

This course will be organized around algorithmic issues that arise in machine learning. The usual paradigm for algorithm design is to give an algorithm that succeeds on all possible inputs, but the difficulty is that almost all of the optimization problems that arise in modern machine learning are *computationally intractable*. Nevertheless, practitioners use a wide variety of heuristics that are successful in practice. However we often do not understand *when* and *why* these approaches work (an issue we would not have if our algorithms came with provable guarantees). The central questions in this course are:

**Question 1** *Which* models *in machine learning lead to tractable algorithmic problems?*

Worst-case analysis is *comfortable* because if an algorithm works in this model, it certainly works in practice. But the optimization problems that machine learning systems "solve" everyday are indeed hard in the worst-case. However these lower bounds are not so frightening; many of the hard instances of machine learning problems are not ones we would want to solve in practice anyways! We will see a number of examples where choosing the right model will lead us to discover new algorithms with provable guarantees, where we really can understand when and why they work. In some cases, we will even be able to analyze approaches that practitioners already use and give new insights into their behavior.

**Question 2** *Can new models – that better represent the instances we actually want to solve in practice – be the inspiration for developing fundamentally new algorithms for machine learning problems? Can we understand when and why widely used heuristics work?*

This course will focus on

(a) nonnegative matrix factorization

(b) topic modeling

(c) tensor decompositions

(d) sparse recovery

(e) dictionary learning

(f) learning mixtures models

(g) matrix completion

Hopefully more sections will be added to this course over time, since there are a vast number of topics at the intersection of algorithms and machine learning left to explore.

# Chapter 2

# Nonnegative Matrix Factorization

In this chapter we will explore the nonnegative matrix factorization problem. We will first recap the motivations from this problem. Next, we give new algorithms that we apply to the classic problem of learning the parameters of a topic model.

## 2.1 Introduction

In order to understand why nonnegative matrix factorization is useful in applications, it will be helpful to compare it to the singular value decomposition. We will focus on applications of both of these to text analysis in this chapter.

### Singular Value Decomposition

Given an $m \times n$ matrix $M$, its singular value decomposition is

$$M = U\Sigma V^T$$

where $U$ and $V$ are orthonormal and $\Sigma$ is diagonal and its entries are nonnegative. Alternatively we can write

$$M = \sum_{i=1}^{r} u_i \sigma_i v_i^T$$

where $u_i$ is the $i^{th}$ column of $U$, $v_i$ is the $i^{th}$ column of $V$ and $\sigma_i$ is the $i^{th}$ diagonal entry of $\Sigma$.

Every matrix has a singular value decomposition! In fact, this representation can be quite useful in understanding the behavior of a linear operator or in general for extracting significant "features" from a large data matrix. We will focus our discussion of the singular value decomposition on the latter. One of the many useful properties of this decomposition is that we can immediately read-off the best low rank approximation to $M$ from it.

**Definition 2.1.1** *The Frobenius norm of a matrix $M$ is $\|M\|_F = \sqrt{\sum_{i,j} M_{i,j}^2}$. Alternately, if $M = \sum_{i=1}^r u_i \sigma_i v_i^T$, $\|M\|_F = \sqrt{\sum \sigma_i^2}$.*

Consider the following optimization problem: Let $B$ be the best rank $k$ approximation to $M$ in the Frobenius norm - i.e. $B$ is the minimizer of $\|M - B\|_F$ over all rank at most $k$ matrices. Then we can without loss of generality choose $B$ to be the first $k$ terms of the singular value decomposition.

**Theorem 2.1.2 (Eckart-Young)** *The best rank $k$ approximation to $M$ in Frobenius norm is attained by $B = \sum_{i=1}^k u_i \sigma_i v_i^T$, and its error is $\|M - B\|_F = \sqrt{\sum_{i=k+1}^r \sigma_i^2}$.*

This is one of the reasons why the singular value decomposition is so widely useful: if we are given data in the form of a matrix $M$ but we believe that the data is approximately low-rank, a natural approach to making use of this structure is to instead work with the best rank $k$ approximation to $M$. This theorem is quite robust and holds even when we change how we measure how good $B$ is as an approximation to $M$:

**Definition 2.1.3** *The operator norm of a matrix $M$ is $\|M\|_2 = \max_{|v|=1} \|Mv\|_2$. Then if $M = \sum_{i=1}^r u_i \sigma_i v_i^T$, $||M||_2 = \sigma_1$ (the largest singular value).*

The best approximation to $M$ in the operator norm is also attained by $B = \sum_{i=1}^k u_i \sigma_i v_i^T$, in which case the error is $\|M - B\|_2 = \sigma_{k+1}$.

Let us give one more interpretation of the singular value decomposition. We can regard an $m \times n$ matrix $M$ as a collection of $n$ data points in $\Re^m$. We associate a distribution $\Delta$ with this set of points which chooses a point uniformly at random. Further suppose that the expectation of this distribution is zero. Our data is in high dimension, and a natural question to ask is: how should we project our data onto a one dimensional subspace in a manner that preserves as much information as possible? One concrete goal is to find a direction $u$ so that projecting $\Delta$ on $u$ maximizes the variance (among all one-dimensional projections). The question leads to another characterization of the singular vectors:

$$u_1 = \text{argmax}\frac{\|u^T M\|_2}{\|u\|_2}$$

and the maximum is $\sigma_1$. Similarly if we want to project onto a two-dimensional subspace so as to maximize the projected variance we should project on span$(u_1, u_2)$. Relatedly

$$u_2 = \min_{u_1}\text{argmax}_{u \perp u_1}\frac{\|u^T M\|_2}{\|u\|_2}$$

and the maximum is $\sigma_2$. This is called the variational characterization of singular vectors. (Here we have assumed the singular values are distinct).

There are efficient algorithms to compute the singular value decomposition. If $n \neq m$ then these algorithms run in time $O(mn^2)$. The basic idea is to reduce $M$ to bidiagonal form using Householder reflections, and then to compute the singular value decomposition from this representation using the QR algorithm. Next we will describe an application to text analysis.

## Applications to Text Analysis

**Latent Semantic Indexing:** [49]

Suppose we are give a large collection of documents, and we would like to extract some hidden structure in this collection (either with the goal of performing information retrieval, or clustering). The usual first step is to collect the data in a very large, very sparse matrix:

**Definition 2.1.4** *The term-by-document matrix $M$ is an $m \times n$ matrix where each row represents a word, each column represents a document and the entry in row $i$, column $j$ is the number of times that word $i$ occurs in document $j$.*

We have clearly lost some information, since this representation does not take into account the order of the words. However matrices are much easier to work with, and the underlying assumption is that it should still be possible to cluster the documents just knowing what words each one contains but not their order. This is often called the *bag-of-words* assumption.

The idea behind latent semantic indexing is to compute the singular value decomposition of $M$ and use this for information retrieval and clustering. More precisely, if we write

$$M \approx U^{(k)}\Sigma^{(k)}V^{(k)T}$$

where $U^{(k)}$ is the first $k$ columns of $U$, etc. then the columns of $U^{(k)}$ are the $k$ directions that maximize the projected variance of a random document. These vectors are interpreted as "topics". More precisely, suppose we want to compute a "similarity" score for document $i$ and document $j$. We could do this by computing

$$\langle M_i, M_j \rangle$$

where $M_i$ is the $i^{th}$ column of $M$, etc. This function "counts" the number of words in common. In particular, given a query we would judge how similar a document is to it just be counting how many of its words occur in each document. This is quite naive. Instead, we could compute

$$\langle M_i^T U^{(k)}, M_j^T U^{(k)} \rangle$$

Intuitively this maps each document to a vector of length $k$ that measures how much of each topic is present in the document, and computes the similarly of the documents by taking an inner-product in this low-dimensional space. In practice this is a much better way to measure similarity and was introduced by the seminal paper of Deerwester et al [49].

However it has its own weaknesses. This approach has some rather undesirable properties:

(a) "topics" are orthonormal

Consider topics like "politics" and "finance". Are the sets of words that describe these topics uncorrelated? No!

(b) "topics" contain negative values

This is more subtle, but negative words can be useful to signal that document is not about a given topic. But then when we compute similarly, two documents are judged to be more similar based on a topic that they are both decidedly not about. This is another counter intuitive and undesirable property.

## Nonnegative Matrix Factorization

The idea due to [73] and [98] is to write

$$M \approx AW$$

where $A$ and $W$ are $m \times k$ and $k \times n$ respectively and are required to be entry-wise nonnegative. In fact, let us suppose that the columns of $M$ each sum to one. It is not hard to see that if $D$ is a diagonal matrix where the $i^{th}$ entry is the reciprocal of the sum of the entries in the $i^{th}$ column of $A$ then $M = \widetilde{A}\,\widetilde{W}$ where $\widetilde{A} = AD$ and $\widetilde{W} = D^{-1}W$ normalizes the data so that the columns of $\widetilde{A}$ and of $\widetilde{W}$ each sum to one. Hence we are finding a set of topics (the columns of $\widetilde{A}$ which are each distributions on words) so that every document can be obtained as a convex combination of the topics that we have found.

This optimization problem plays a crucial role in many machine learning systems, such as image segmentation, text analysis, recommendation systems, etc. But this optimization problem is $NP$-hard [115]. So what should we do now? Give up?

In contrast, singular value decomposition is a problem where theory and practice agree! It can be computed efficiently, and it has many uses. But in spite of this intractability result, nonnegative matrix factorization really is used in practice. The standard approach is to use *alternating minimization*:

**Alternating Minimization:** This problem is non-convex, but suppose we guess $A$. Then computing the nonnegative $W$ that minimizes $\|M - AW\|_F$ is convex and can be solved efficiently. The approach is to guess $A$, compute the best $W$ then set $W$ as fixed and compute the best $A$, and so on. This process converges, but not necessarily to the optimal solution.

*It can and does get stuck in local minima in practice!*

We note that this approach is also called *expectation-maximization* [50], and is the standard approach not just for nonnegative matrix factorization, but for many other problems we will study in this course such as *dictionary learning* and *learning mixtures models.*

## Food for Thought

But maybe heuristics like this are identifying interesting instances of the problem. The goal of this course is to not give up when faced with intractability, and to look for new explanations. These explanations could be new models (that avoid the aspects of the problem that allow us to embed hard problems) or could be identifying conditions under which heuristics that are already used, do work. This is a largely unexplored area.

In the next section, we will ask what happens if we restrict the number of topics. The instances generated by [115] have $k$ linear in $m$ and $n$, but when we look for a set of topics that explain $300,000$ New York Times articles, we are looking for only a few hundred topics. So one way to reformulate the question is to ask what its complexity is as a function of $k$. We will essentially resolve this using algebraic techniques. Nevertheless if we want even better algorithms, we need more assumptions. We will see how a geometric interpretation of this problem implies that these hard instances are unstable, and we will examine a condition (separability) that enforces stability, and allows us to give much better algorithms - ones that run in time polynomial in all of the parameters.

## 2.2 Algebraic Algorithms

In the previous section we introduced the nonnegative matrix factorization problem and described its applications to text analysis (it has many other applications). Vavasis proved that this problem is $NP$-hard in the worst-case, but the instances he contracted have $k$ – the number of topics – linear in the size of the matrix [115]. In most practical applications, $k$ is much smaller than $m$ or $n$ and with this in mind we will instead ask: What is the complexity of this problem as a function of $k$? We will make use of tools from algebra to give a polynomial time algorithm for any $k = O(1)$. In fact, the algorithm we present here will be nearly optimal in terms of its dependence on $k$.

### Definitions

Let us define the nonnegative matrix factorization problem formally, since we did so only informally in the previous section: Suppose we are given an entry-wise nonnegative matrix $M$ of size $m \times n$.

**Definition 2.2.1** *The nonnegative rank of $M$ – denoted by $rank^+(M)$– is the smallest $k$ such that there are nonnegative matrices $A$ and $W$ of size $m \times k$ and $k \times n$ respectively that satisfy $M = AW$.*

Equivalently, $\text{rank}^+(M)$ is the smallest $k$ such that there are $k$ nonnegative rank one matrices $\{M_i\}$ that satisfy $M = \sum_i M_i$.

Both of these equivalent formulations of the problem will be useful throughout our discussion. To gain some familiarity with this parameter, it is helpful to compare it to a more familiar one: If we omit the requirement that $A$ and $W$ be entry-wise nonnegative, then the smallest $k$ is precisely the rank of $M$. Hence the following relation is immediate:

**Fact 2.2.2** *$rank^+(M) \geq rank(M)$*

In fact the rank and the nonnegative rank of a matrix can be quite different:

*Example.* Let $M \in \mathbb{M}_{n\times n}$, where $M_{ij} = (i-j)^2$. It is easy to see that the columns of $M$ are spanned by

$$\left\{ \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ \vdots \\ n \end{bmatrix}, \begin{bmatrix} 1^2 \\ 2^2 \\ \vdots \\ n^2 \end{bmatrix} \right\}$$

It is easy to see that $\text{rank}(M) = 3$ However, $M$ has zeros along the diagonal and non-zeros off it. Furthermore for any rank one nonnegative matrix $M_i$, its pattern of zeros and non-zeros is a *combinatorial rectangle* - i.e. the intersection of some set of rows and columns - and a standard argument implies that $\text{rank}^+(M) = \Omega(\log n)$. There are examples with even larger separations too.

Next we will connect nonnegative matrix factorization to computational problems involving systems of polynomial inequalities.

## Systems of Polynomial Inequalities

We can reformulate the problem of finding an $A$ and $W$ that prove $\text{rank}^+(M) \leq k$ as a problem of finding a feasible solution to a particular system of polynomial inequalities. More specifically, the problem we want to solve is:

$$\begin{cases} M & = AW \\ A & \geq 0 \\ W & \geq 0 \end{cases} \tag{2.1}$$

This system consists of quadratic equality constraints (one for each entry of $M$), and linear constraints that $A$ and $W$ be entry-wise nonnegative. Before trying to design better algorithms for $k = O(1)$, we should ask a more basic question (whose answer is not at all obvious):

**Question 3** *Is there any finite time algorithm?*

The difficulty is that even if there is a solution, the entries of $A$ and $W$ could be irrational. This is quite different than, say, 3-SAT where there is a simple brute-force algorithm. In contrast for nonnegative matrix factorization it is quite challenging to design algorithms that run in any finite amount of time. But indeed there are algorithms (that run in some fixed amount of time) to decide whether a system of polynomial inequalities has a solution or not in the real RAM model. These algorithms can also compute an implicit representation of the solution, if there is one. The output is a polynomial and an interval (for each variable) in which there is only one root, which is the value of the variable in the true solution. And you can find as many bits of the solution as you would like by performing binary search for the root.

The first algorithm follows from the seminal work of Tarski, and there has been a long line of improvements based on successively more powerful algebraic decompositions. This line of work culminated in algorithms whose running time is exponential in the number of variables but is polynomial in all the other parameters of the problem (the number of polynomial inequalities, the maximum degree and the bit complexity of the coefficients). The running time is $(nD)^{O(r)}$ where $n$ is the number of polynomial inequalities, $D$ is the maximum degree and $r$ is the number of variables [106]. This running time is essentially optimal under the exponential time hypothesis [78]. In particular, if there is an algorithm for this problem that runs in time $(pD)^{o(r)}$ then it would yield sub-exponential time algorithms for 3-SAT.

We can use these algorithms to solve nonnegative matrix factorization. However the number of variables we would need in the naive representation is $nk + mk$, one for each entry in $A$ or $W$. So even if $k = O(1)$, we would need a linear number of variables and the running time would be exponential. However we could hope that even though the naive representation uses many variables, perhaps there is a more clever representation that uses many fewer variables. Can we reduce the number of variables in the system of polynomial inequalities from $O(nk + mk)$ to $f(k)$?

If we could do this, then we could solve nonnegative matrix factorization in polynomial time for any $k = O(1)$. Next, we will describe some basic tools in the first-order theory of the reals. These results will help formalize our intuition from above that the number of variables is the right complexity measure when reasoning about how difficult it is to solve a system of polynomial inequalities, but their proof is out of scope of this course.

## First-Order Theory of the Reals

**Definition 2.2.3** *A set $S$ is semialgebraic if there exist multivariate polynomials $p_1, ..., p_n$ such that*

$$S = \{x_1, ..., x_r | p_i(x_1, ..., x_r) \geq 0\}$$

*or if $S$ is a finite union or intersection of such sets.*

**Definition 2.2.4** *The projection of a semialgebraic set $S$ is defined as*

$$proj_S(X_1, ..., X_\ell) = \{x_1, ..., x_\ell | \exists\, x_{\ell+1}, ..., x_r \text{ such that } p(x_1, ..., x_r) \in S\}$$

**Theorem 2.2.5 (Tarski)** *The projection of a semialgebraic set is semialgebraic.*

This is one of the foundational results in the field, and is often called *quantifier elimination* [110], [107]. To gain some familiarity with this notion, consider the case of algebraic sets (defined analogously as above, but with polynomial equality constraints instead of inequalities). Indeed, the above theorem implies that the projection of an algebraic set is itself semi-algebraic. Is its projection also algebraic? No (e.g. think about the projection of a circle)!

Earlier, we stated that there are algorithms to solve systems of polynomial inequalities (and find an implicit representation for the solution, if there is one) in time $(nD)^{O(r)}$ where $n$ is the number of polynomial inequalities, $D$ is the maximum degree and $r$ is the number of variables [106]. In fact, these algorithms work in a more general setting where there is additionally a boolean function $\mathbb{B}$ that constraints the sign pattern of the polynomials. We are interested in deciding whether the set

$$S = \{x_1, ..., x_r | \mathbb{B}(p_1(x_1, ..., x_r), ..., p_n(x_1, ..., x_r)) = \text{true}\}$$

is non-empty, and we assume that we can evaluate $\mathbb{B}$ (but not, say, that it has a succinct circuit). A related result is the famous Milnor-Warren bound (see e.g. [7]):

**Theorem 2.2.6 (Milnor-Warren)** *Given $n$ polynomials $p_1, ..., p_m$ of degree $\leq D$ on $r$ variables $\mathbf{x} = x_1, ...x_r$, consider the sign pattern at $\mathbf{x}$:*

$$\mathbf{x} \to \big(sgn(p_1(\mathbf{x})), sgn(p_2(\mathbf{x})), ..., sgn(p_m(\mathbf{x}))\big)$$

*Then as $\mathbf{x}$ ranges over $\mathbb{R}^r$ the number of distinct sign patterns is at most $(nD)^r$.*

A priori we could have expected as many as $3^n$ sign patterns. In fact, algorithms for solving systems of polynomial inequalities are based on cleverly enumerating the set of sign patterns so that the total running time is dominated by the maximum number of distinct sign patterns that there could be! In fact, the Milnor-Warren bound can be thought of as an analogue of the Sauer-Shelah lemma that is used throughout supervised learning where the number of variables plays the role of the $VC$-dimension.

Next we will give a technique to reduce the number of variables.

## Variable Reduction

It is clear that the set of points satisfying (2.1) is a semialgebraic set. However even for $k = 3$ this system has a linear (in $n$ and $m$) number of variables, so directly solving (2.1) would require exponential time.

**Question 4** *Can we find an alternate system of polynomial inequalities that expresses the same decision problem but uses many fewer variables?*

We will focus on a special case called *simplicial factorization* where $\text{rank}(M) = k$. In this case, we are asking whether or not $\text{rank}^+(M) = \text{rank}(M) = k$ and this simplifies matters because of the following observation:

**Claim 2.2.7** *In any solution, $A$ and $W$ must have full column and row rank respectively.*

**Proof:** The span of the columns of $A$ must contain the columns of $M$ and similarly the span of the rows of $W$ must contain the rows of $M$. Since $\text{rank}(M) = k$ and $A$ and $W$ have $k$ columns and rows respectively we conclude that the $A$ and $W$ must have full column and row rank respectively. Moreover their span must be the column space and row space of $M$ respectively. ■

Hence we know that $A$ and $W$ have left and right pseudo-inverses $A^+$ and $W^+$ respectively. We will make use of these pseudo-inverses to reduce the number of variables in our system of polynomial inequalities: We have that $A^+A = I_r$ where $I_k$ is the $k \times k$ identity. Hence

$$A^+AW = W$$

and so we can recover the columns of $W$ from a linear transformation of the columns of $M$. This leads to the following alternative system of polynomial inequalities:

$$\begin{cases} MW^+A^+M & = M \\ MW^+ & \geq 0 \\ A^+M & \geq 0 \end{cases} \tag{2.2}$$

A priori, it is not clear that we have made progress since this system also has $nk + mk$ variables corresponding to the entries of $A^+$ and $W^+$. However consider the matrix $A^+M$. If we represent $A^+$ as an $k \times n$ matrix then we are describing its action on all vectors, but the crucial observation is that we only need to know how $A^+$ acts on the columns of $M$ which span a $k$ dimensional space. Hence we can apply a change of basis to rewrite $M$ as $M_R$ which is an $k \times m$ matrix, and there is an $k \times k$ linear transformation $T$ (obtained from $A^+$ and the change of basis) so that $TM_R = W$. A similar approach works for $W$, and hence we get a new system:

$$\begin{cases} M_CSTM_R & = M \\ M_CS & \geq 0 \\ TM_R & \geq 0 \end{cases} \tag{2.3}$$

The variables of this system are the entries in $S$ and $T$. So there are $2k^2$ variables. And the properties we need of this system are that

(a) If the simplicial factorization problem has a solution, then there is a solution to this system (completeness)

(b) If there is any solution to the system, then the simplicial factorization has a solution (soundness)

We have already proven the first property, and the second property follows because we can set $A = M_C S$ and $W = TM_R$ and this is a valid factorization with inner-dimension $k$. Hence if we apply Renegar's algorithm to this new system, the algorithm runs in time $(nm)^{O(k^2)}$ and solves the simplicial factorization problem.

The above approach is based on the paper of Arora et al [13] where the authors also give a variable reduction procedure for nonnegative matrix factorization (in the general case where $A$ and $W$ need not have full column or row rank respectively). The authors reduce the number of variables from $(nk + mk)$ to $f(k) = 2k^2 2^k$ and this yields a doubly-exponential time algorithm as a function of $k$. The crucial observation is that even if $A$ does not have full column rank, we could write a system of polynomial inequalities that has a pseudo-inverse for each set of its columns that is full rank (and similarly for $W$). However $A$ could have as many as $\binom{k}{k/2}$ maximal sets of linearly independent columns, and hence the resulting system of polynomial inequalities has $f(k)$ variables but $f(k)$ is itself exponential in $k$.

In [94] the author further reduces the number of variables to $2k^2$ for nonnegative matrix factorization, and the main idea is that even though $A$ could have exponentially many maximal sets of linearly independent columns, their psueudo-inverses are algebraically dependent and can be expressed over a common set of $k^2$ variables using Cramer's rule. This yields a singly exponential time algorithm for nonnegative matrix factorization that runs in $(nm)^{O(k^2)}$ time which is essentially optimal since any algorithm that runs in time $(nm)^{o(k)}$ would yield a sub-exponential time algorithm for 3-SAT [13].

## 2.3 Stability and Separability

In the previous section we took an algebraic approach and here instead we will work with an assumption called *separability* [54] which will allow us to give an algorithm that runs in polynomial time (even for large values of $r$). Our discussion will revolve around the intermediate simplex problem.

## Intermediate Simplex Problem

Let us define the intermediate simplex problem:

> We are given two polytopes $Q$ and $P$ with $P \subseteq Q$ and furthermore $P$ is encoded by its vertices and $Q$ is encoded by its facets. Is there a simplex $K$ with $P \subseteq K \subseteq Q$?

We would like to connect this problem to nonnegative matrix factorization, since it will help us build up a geometric view of the problem. Consider the following problem:

> Given nonnegative matrices $M$ and $A$, does there exists $W \geq 0$ such that $M = AW$?

The answer is "Yes", if and only if each column of $M$ is in the cone spanned by nonnegative combinations of the columns of $A$. Moreover if we normalize the columns of $M$ and $A$ so that they sum to one, then the answer is "Yes" if and only if the convex hull of the columns of $A$ contains the columns of $M$. Recall in simplicial factorization we are given a nonnegative matrix $M$ with $\text{rank}(M) = k$, and our goal is to decide whether or not $\text{rank}^+(M) = k$. We will prove that the simplicial factorization problem and the intermediate simplex problem are equivalent [115]. Consider the following helper problem, which we call **(P0):**

> Given $M = UV$, is there an invertible $k \times k$ matrix $T$ such that $UT^{-1}$, and $TV$ are nonnegative?

In fact, Vavasis [115] proved that **(P0)**, intermediate simplex and the simplicial factorization problem are each polynomial time interreducible. It is easy to see that **(P0)** and the simplicial factorization problem are equivalent since in any two factorizations $M = UV$ or $M = AW$ (where the inner-dimension equals the rank of $M$), the column spaces of $M$, $U$ and $A$ are identical. Similarly the rows spaces of $M$, $V$ and $W$ are also identical.

The more interesting aspect of the proof is the equivalence between **(P0)** and the intermediate simplex problem. The translation is:

(a) rows of $U \iff$ vertices of $P$

(b) rows of $T \iff$ vertices of $K$

(c) columns of $V \iff$ facets of $Q$

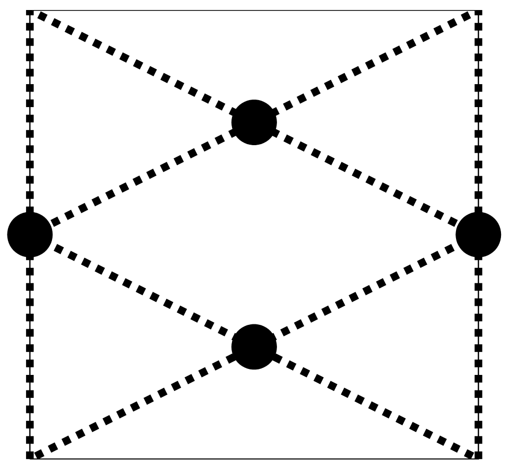

Figure 2.1: This figure is taken from [115]. The intermediate simplex problem has two solutions which will be used to encode the truth assignment of a variable.

Then the constraint that $UT^{-1}$ is nonnegative is (roughly) the constraint that $P \subseteq K$ and the constraint $TV$ is (roughly) the constraint $K \subseteq Q$. There are some tedious normalization issues that arise since we need to be careful about the distinction between the convex hull of a set of vectors and the cone generated by all nonnegative combinations. However this equivalence gives us a geometric view that will be helpful.

Vavasis made use of the equivalences in the previous subsection to prove that nonnegative matrix factorization is $NP$-hard. Consider the gadget in Figure 2.1; the crucial property is that there are only two possible intermediate triangles, which can then be used to represent the truth assignment for a variable $x_i$. The description of the complete reduction, and the proof of its soundness are involved (see [115]).

The trouble is that gadgets like those in Figure **??** are *unstable. We can change the number of solutions by small perturbations to the problem.* Motivated by issues of uniqueness and robustness, Donoho and Stodden [54] introduced a condition called *separability* that alleviates many of these problems, which we will discuss in the next subsection.

## Separability

**Definition 2.3.1** *We call $A$* separable *if, for every column of $A$, there exists a row of $A$ whose only non-zero entry is in that column.*

Furthermore in the separable nonnegative matrix factorization problem we are given $M$ and the promise that if there is a nonnegative matrix factorization, there is one in which $A$ is separable. Donoho and Stodden used this condition (and others) to show that there are somewhat natural conditions under which the nonnegative matrix factorization is unique. Arora, Ge, Kannan and Moitra gave an algorithm for finding it:

**Theorem 2.3.2** *[13] Given a nonnegative matrix $M$ with the promise that there is a nonnegative matrix factorization $M = AW$ where $A$ is separable, there is a polynomial time algorithm to compute such a factorization of minimum inner-dimension.*

In fact, separability is quite natural in the context of text analysis. Recall that we interpret the columns of $A$ as topics. We can think of separability as the promise that these topics come with *anchor words*; informally, for each topic there is an unknown anchor word that if it occurs in a document, the document is (partially) about the given topic. For example, *401k* could be an anchor word for the topic *personal finance*.

Why do anchor words help? It is easy to see that if $A$ is separable, then the rows of $W$ appear as rows of $M$ (after scaling). Hence we just need to determine which rows of $M$ correspond to anchor words. We know from our discussion in Section 2.3 that (if we scale $M$, $A$ and $W$ so that their rows sum to one) the convex hull of the rows of $W$ contain the rows of $M$. But since these rows appear in $M$ as well, we can try to find $W$ by iteratively deleting rows of $M$ that do not change its convex hull.

Let $M^i$ denote the $i$th row of $M$ and let $M^I$ denote the restriction of $M$ to the rows in $I$ for $I \subseteq [n]$. So now we can find the anchor words using the following simple procedure:

**Find Anchors [13]**

```
Input: matrix $M \in \mathbb{R}^{n \times m}$ satisfying the conditions in Theorem 2.3.2
Output: $W = M^I$

Set $I = [n]$
For $i = 1, 2, ..., n$
    If $M^i \in \text{conv}(\{M^j | j \in I, j \neq i\})$, set $I \leftarrow I - \{i\}$
End
```

It is easy to see that deleting a row of $M$ that is not an anchor word will not change the convex hull of the remaining rows, and so the above algorithm terminates with a set $I$ that only contains anchor words. Moreover at termination

$$\text{conv}(\{M^i | i \in I\}) = \text{conv}(\{M^j\}_j)$$

Alternatively the convex hull is the same as at the start. Hence the anchor words that are deleted are redundant and we could just as well do without them.

```
Separable NMF [13]
Input: matrix $M \in \mathbb{R}^{n \times m}$ satisfying the conditions in Theorem 2.3.2
Output: $A, W$

Run Find Anchors on $M$, let $W$ be the output
Solve for nonnegative $A$ that minimizes $\|M - AW\|_F$ (convex programming)
End
```

The proof of theorem follows immediately from the proof of correctness of **Find Anchors** and the fact that $\text{conv}(\{M^i\}_i) \subseteq \text{conv}(\{W^i\}_i)$ if and only if there is a nonnegative $A$ (whose rows sum to one) with $M = AW$.

The above algorithm when naively implemented would be prohibitively slow. Instead, there have been many improvements to the above algorithm [27], [84] [65], and we will describe one in particular that appears in [12]. Suppose we choose a row $M^i$ at random. Then it is easy to see that the furthest row from $M^i$ will be an anchor word.

Similarly, if we have found one anchor word the furthest row from it will be another anchor word, and so on. In this way we can greedily find all of the anchor rows, and moreover this method only relies on pair-wise distances and projection so we can apply dimension reduction before running this greedy algorithm. This avoids linear programming altogether in the first step in the above algorithm, and the second step can also be implemented quickly because it involves projecting a point into an $k-1$-dimensional simplex.

## 2.4 Topic Models

Here we will consider a related problem called *topic modeling*; see [28] for a comprehensive introduction. This problem is intimately related to nonnegative matrix factorization, with two crucial differences. Again there is some factorization $M = AW$ but now we do not get access to $M$ but rather $\widetilde{M}$ which is a very crude approximation. Intuitively, each column in $M$ is a document that is itself a distribution on words. But now the words that we observe are samples from this distribution (so we do not actually know the columns of $M$).

The second difference is that we think of $W$ as stochastically generated. There are in fact many popular choices for this distribution:

(a) **Pure Documents:** Each document is about only one topic, hence each column of $W$ has exactly one non-zero.

(b) **Latent Dirichlet Allocation [30] :** The columns of $W$ are generated from a Dirichlet distribution.

(c) **Correlated Topic Model [29] :** Certain pairs of topics are allowed to be positively or negatively correlated, and the precise distribution that generates the columns of $W$ is log-normal.

(d) **Pachinko Allocation Model [89] :** This is a multi-level generalization of LDA that also allows for certain types of structured correlations.

There are many more choices. Regardless, our goal is to learn $A$ from $\widetilde{M}$. To emphasize the differences, note that *even if we knew* $A$ we cannot compute $W$ exactly. Alternatively, $\widetilde{M}$ and $M$ can be quite different since the former may be sparse while the latter is dense. Are there provable algorithms for topic modeling?

## The Gram Matrix

We will follow an approach of Arora, Ge and Moitra [14]. At first this seems like a fundamentally different problem than the ones we have considered because in this model we cannot ask for longer documents, we can only ask for more of them. Hence we are increasing the number of columns of $\widetilde{M}$ but each column is not that close to the corresponding column in $M$. The basic idea is to work instead with the *Gram matrix* $G$:

**Definition 2.4.1** *Let $G$ denote the word $\times$ word matrix whose entry in $(a, b)$ is the probability that the first two words in a randomly chosen document are $a$ and $b$ respectively.*

**Definition 2.4.2** *Let $R$ denote the topic $\times$ topic matrix whose entry in $(i, j)$ is the probability that the first two words (again in a randomly chosen document) are generated from the topics $i$ and $j$ respectively.*

Note that we can approximate $G$ from our samples, however we cannot (directly) approximate $R$ and it is controlled by the choice of which distribution we use to generate the columns of $W$. More precisely:

**Lemma 2.4.3** $G = ARA^T$

**Proof:** Let $w_1$ denote the first word and let $t_1$ denote the topic of $w_1$ (and similarly for $w_2$). We can expand $\mathbb{P}[w_1 = a, w_2 = b]$ as:

$$\sum_{i,j} \mathbb{P}[w_1 = a, w_2 = b | t_1 = i, t_2 = j] \mathbb{P}[t_1 = i, t_2 = j]$$

and the lemma is now immediate. ■

The key observation is that $G$ has a separable nonnegative matrix factorization given by $A$ and $RA^T$ since $A$ is separable and the latter matrix is nonnegative. Indeed if $RA^T$ has full row rank then the algorithm in Theorem 2.3.2 will find the true set of anchor words. However since the rows of $RA^T$ are no longer normalized to sum to one, the above factorization is not necessarily unique. Nevertheless we have made some progress, and we can adopt a Bayesian interpretation (see [12]).

## Recovery via Bayes Rule

In fact, the entries of $A$ are conditional probabilities $\mathbb{P}(w_1|t_1)$ and so we can reason about the posterior distribution $\mathbb{P}(t_1|w_1)$. In fact this gives us an alternate characterization of an anchor word: A word is an anchor word if and only if its posterior distribution is supported on just one topic. In particular

$$\mathbb{P}(t_1 = t | w_1 = w) = \begin{cases} 1, & w \text{ is an anchor word for } t, \\ 0, & \text{otherwise}, \end{cases}$$

Now we can expand:

$$P(w_1 = w' | w_2 = w) = \sum_t \mathbb{P}(w_1 = w' | w_2 = w,\, t_2 = t) \cdot \mathbb{P}(t_2 = t | w_2 = w),$$

In fact $w_1$ is independent of $w_2$ if we condition on $t_2$ and so:

$$\begin{aligned} \mathbb{P}(w_1 = w' | w_2 = w,\, t_2 = t) &= \mathbb{P}(\text{word1} = w' | \text{topic2} = t) \\ &= \mathbb{P}(\text{word1} = w' | \text{word2} = \text{anchor}(t)), \end{aligned}$$

which we can compute from $G$ after having determined the anchor words. Hence:

$$\mathbb{P}(w_1 = w' | w_2 = w) = \sum_t \mathbb{P}(\text{word1} = w' | \text{word2} = \text{anchor}(t)) \mathbb{P}(t_2 = t | w_2 = w)$$

which we can think of a linear systems in the variables $\{\mathbb{P}(t_2 = t | w_2 = w)\}$. It is not hard to see that if $R$ has full rank then it has a unique solution. Finally, we compute the probabilities we were originally interested in by Bayes' rule:

$$\begin{aligned}\mathbb{P}(\text{word } w | \text{topic } t) &= \frac{\mathbb{P}(\text{topic } t | \text{word } w) \cdot \mathbb{P}(\text{word } w)}{\mathbb{P}(\text{topic } t)} \\ &= \frac{\mathbb{P}(\text{topic } t | \text{word } w) \cdot \mathbb{P}(\text{word } w)}{\sum_{w'} \mathbb{P}(\text{topic } t | \text{word } w') \cdot \mathbb{P}(\text{word } w')}.\end{aligned}$$

We can now state the algorithm **Recover**. Let $\widetilde{G}$ be the empirical Gram matrix, where $\widetilde{G}_{a,b}$ is the fraction of documents in our sample whose first word is $a$ and whose second word is $b$.

Suppose each anchor word has probability at least $p$. Then the main result in this subsection is:

**Theorem 2.4.4** *[14] For any separable topic model where $R$ is full rank there is a polynomial time algorithm to compute $\widetilde{A}$ that is $\varepsilon$-close to $A$ and the running time and sample complexity (number of documents) is poly$(n, 1/p, 1/\varepsilon, 1/\sigma_{min}(R))$, provided documents have length at least two.*

In the next subsection we describe some experimental results.

```
**Recover [14], [12]**
Input: term-by-document matrix $M \in \mathbb{R}^{n \times m}$
Output: $A, R$

Compute $\widetilde{G}$, compute $\mathbb{P}(w_1 = w | w_2 = w')$
Run **Find Anchors**
Solve for $\mathbb{P}(\text{topic } t | \text{word } w)$ and use Bayes' rule to compute $A$
End
```

## Experiments

We are faced with a basic *scientific* question now: Are there really anchor words? The following experiment was conducted in [12]:

(a) Run MALLET (a popular topic modeling toolkit) on a collection of New York Times articles, its output is a topic matrix $A$.

(b) Use $A$ to generate data from a topic model, run MALLET on this data.

The important point is that here the data that we are running on is actually from a topic model and we can compare how well one algorithm can recover the true matrix compared to how well another algorithm does. Then:

(c) Run the new algorithm on this data.

This is a seemingly unfair comparison, since we have restricted ourselves to a topic matrix $A$ that MALLET has already found once (so this is our notion of what constitutes a realistic topic model). Yet surprisingly the algorithm in the previous subsection was able to find the topic matrix $A$ more accurately and orders of magnitude faster! *This is an important example where finding conditions under which we can give provable algorithms indeed led to much better algorithms in practice.*

# Chapter 3

# Tensor Methods

In this chapter we will study algorithms for tensor decompositions and their applications to statistical inference.

## 3.1 Basics

Here we will introduce the basics of tensors. A matrix is an *order* two tensor – it is indexed by a pair of numbers. In general a tensor is indexed over $k$-tuples, and $k$ is called the *order* of a tensor. We can think of a tensor $T$ as a point in $\mathbb{R}^{n_1 \times n_2 \times \ldots \times n_k}$. We will mostly be interested in order three tensors throughout this chapter. If $T$ is an order three tensor of size $m \times n \times p$ we can regard $T$ as a collection of $p$ matrices of size $m \times n$ that are stacked on top of each other.

We can generalize many of the standard definitions from linear algebra to the tensor setting, however we caution the reader that while these parameters are easy to compute for matrices, most parameters of a tensor are hard to compute (in the worst-case).

**Definition 3.1.1** *A rank one tensor is a tensor of the form $T = u \otimes v \otimes w$ where $T_{i,j,k} = u_i v_j w_k$. And in general the rank of a tensor $T$ is the minimum $r$ such that we can write $T$ as the sum of $r$ rank one tensors.*

**Question 5** *Tensors are computationally more difficult to work with; so why should we try to work with them?*

In fact, we will give a motivating example in the next section that illustrates the usefulness of tensor methods in statistics and machine learning (and where matrices are not sufficient).

## Case Study: Spearman's Hypothesis

Charles Spearman was a famous psychologist who postulated that there are essentially two types of intelligence: *mathematical* and *verbal*. In particular, he believed that how well a student performs at a variety of tests depends only on their intrinsic aptitudes along these two axes. To test his theory, he set up a study where a thousand students each took ten various types of test. He collected these results into a matrix $M$ where the entry $M_{i,j}$ was used to denote how well student $i$ performed on test $j$. Spearman took the best rank two approximation to $M$. In other words, that there exists vectors (not necessarily unit vectors) $u_1, u_2 \in \mathbb{R}^{1000}$, $v_1, v_2 \in \mathbb{R}^{10}$, such that

$$M \approx u_1 v_1^T + u_2 v_2^T$$

This is called *factor analysis*, and his results somewhat confirmed his hypothesis. But there is a fundamental obstacle to this type of approach that is often referred to as the "Rotation Problem". Set $U = [u_1, u_2]$ and $V = [v_1, v_2]$ and let $O$ be an orthogonal matrix. Then

$$UV^T = UO\ O^T V^T$$

is an alternative factorization that approximates $M$ just as well. However the columns of $UO$ and the rows of $O^T V^T$ could be much less interpretable. To summarize, just because there is a good factorization of a given data matrix $M$ does not mean that factor analysis will find it.

Alternatively, suppose we are given a matrix $M = \sum_{i=1}^{r} x_i y_i^T$.

**Question 6** *Can we determine* $\{x_i\}_i$ *and* $\{y_i\}_i$ *if we know* $M$*?*

Actually, there are only trivial conditions under which we can uniquely determine these factors. If $r = 1$ of if we know for a priori reasons that the vectors $\{x_i\}_i$ and $\{y_i\}_i$ are orthogonal, then we can. But in general we could take the singular value decomposition of $M = U\Sigma V^T$ and take $\{\sigma_i u_i\}_i$ and $\{v_i\}_i$ to be an alternative set of factors that explain $M$ (and if $\{x_i\}_i$ and $\{y_i\}_i$ are not orthogonal, then these are clearly two different sets of factors for the same $M$).

However if we are given a tensor

$$T = \sum_{i=1}^{r} x_i \otimes y_i \otimes w_i$$

then there are general conditions (namely if $\{x_i\}_i$, $\{y_i\}_i$ and $\{w_i\}_i$ are each linearly independent) not only is the true factorization the unique factorization of $T$ with rank $r$ but in fact there are simple algorithms to find it! This is precisely the reason that tensor methods are ubiquitous in statistics and machine learning: If we are given a tensor whose factors represent the parameters of a statistical model, we can find these factors efficiently; yet for matrices the factors are not uniquely determined.

## Complexity of Tensor Problems

In the previous subsection, we alluded to the fact that tensor methods will offer a way around the "Rotation Problem" which is a common obstacle in factor analysis. So can we just compute the minimum rank decomposition of a tensor? In fact, not only is this problem computationally hard (without further assumptions) but most tensor problems are hard [71]! Even worse, many of the standard relations in linear algebra do not hold and even the definitions are in some cases not well-defined.

(a) For a matrix $A$, $\dim(\text{span}(\{A_i\}_i)) = \dim(\text{span}(\{A^j\}_j))$ (the column rank equals the row rank).

However no such relation holds for tensors.

(b) For a matrix $A$, the best rank $k$ approximation to $A$ can be obtained from its best rank $k+1$ approximation.

In particular, if we let $A^{(k+1)}$ be the best rank $k+1$ approximation to $A$, then the best rank $k$ approximation to $A^{(k+1)}$ is the best rank $k$ approximation to $A$. But for tensors the best rank $k$ and rank $k+1$ approximations do not necessarily share any common rank one factors. In fact, subtracting the best rank one approximation to a tensor $T$ from it can actually increase its rank.

(c) For a real-valued matrix its rank over $\mathbb{R}$ and over $\mathbb{C}$ are the same, but this is false for tensors.

There are real-valued tensors whose minimum rank decomposition requires complex numbers.

Perhaps the most worrisome issue is that in some cases the definitions fail too:

**Definition 3.1.2** *The* border rank *of a tensor $T$ is the minimum $r$ such that for any $\varepsilon > 0$ there is a rank $r$ tensor that is entry-wise $\varepsilon$ close to $T$.*

We remark that what norm we use in the above definition is not important. In fact, for matrices the border rank is equal to the rank. But for tensors these can be different.

(d) For a tensor, its border rank is not necessarily equal to its rank.

Consider the following $2 \times 2 \times 2$ tensor $T$, over $\mathbb{R}$:

$$T = \left( \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \right).$$

We will omit the proof that $T$ has rank 3, but show that $T$ admits an arbitrarily close rank 2 approximation. Consider the following matrices

$$S_n = \left( \begin{pmatrix} n & 1 \\ 1 & \frac{1}{n} \end{pmatrix}, \begin{pmatrix} 1 & \frac{1}{n} \\ \frac{1}{n} & \frac{1}{n^2} \end{pmatrix} \right) \text{ and } R_n = \left( \begin{pmatrix} n & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \right).$$

It is not too hard to see that $S_n = n \binom{1}{1/n} \otimes \binom{1}{1/n} \otimes \binom{1}{1/n}$, and hence is rank 1, and $R_n$ is also rank 1. Thus the tensor $S_n - R_n$ is rank 2, but also is an $1/n$ entry-wise approximation of $T$.

One last issue is that it is easy to see that a random $n \times n \times n$ tensor will have rank $\Omega(n^2)$, but it is unknown how to explicitly construct any order three tensor whose rank is $\Omega(n^{1+\varepsilon})$. And any such construction would give the first super-linear circuit lower bounds for any explicit problem [102] which is a long-standing open question in circuit complexity.

## Jennrich's Algorithm

While we cannot hope for algorithms that find the minimum rank decomposition of a tensor in general, in fact there are mild conditions under which we can do it. This algorithm has been rediscovered numerous times in a wide range of applications, and after an extensive search we discovered that this simple algorithm was first reported in a working paper of Harshman [70] where the author credits Dr. Robert Jennrich. We will state and prove a version of this result that is more general, following the approach of Leurgans, Ross and Abel [87]:

**Theorem 3.1.3** *[70], [87] Consider a tensor*

$$T = \sum_{i=1}^{r} u_i \otimes v_i \otimes w_i$$

*where each set of vectors $\{u_i\}_i$ and $\{v_i\}_i$ are linearly independent, and moreover each pair of vectors in $\{w_i\}_i$ are linearly independent too. Then the above decomposition is unique up to rescaling, and there is an efficient algorithm to find it.*

We will see a wide variety of applications of this basic result (which may explain why it has been rediscovered so many times) to phylogenetic reconstruction [96], topic modeling [8] and community detection [9]. This decomposition also plays a crucial role in learning mixtures of spherical Gaussians [75] and independent component analysis [36], although we will instead present a local search algorithm for the latter problem.

```
**Tensor Decomposition [70], [87]**
Input: tensor $T \in \mathbb{R}^{m \times n \times p}$ satisfying the conditions in Theorem 3.1.3
Output: factors $\{u_i\}_i$, $\{v_i\}_i$ and $\{w_i\}_i$

Choose $a, b \in \mathbb{S}^{p-1}$ uniformly at random; set $T_a = T(*,*,a)$ and $T_b = T(*,*,b)$

Compute the eigendecomposition of $T_a(T_b)^+$ and $T_b(T_a)^+$
Let $U$ and $V$ be the eigenvectors
Pair up $u_i$ and $v_i$ iff their eigenvalues are reciprocals

Solve for $w_i$ in $T = \sum_{i=1}^r u_i \otimes v_i \otimes w_i$
End
```

Recall that $T_a$ is just the weighted sum of matrix slices through $T$, each weighted by $a_i$. It is easy to see that:

**Claim 3.1.4** *$T_a = \sum_{i=1}^r \langle w_i, a\rangle u_i v_i^T$ and $T_b = \sum_{i=1}^r \langle w_i, b\rangle u_i v_i^T$*

Alternatively, let $D_a = \text{diag}(\{\langle w_i, a\rangle\}_i)$ and let $D_b = \text{diag}(\{\langle w_i, b\rangle\}_i)$. Then we can write $T_a = U D_a V^T$ and $T_b = U D_b V^T$ where the columns of $U$ and $V$ are $u_i$ and $v_i$ respectively.

**Lemma 3.1.5** *The eigenvectors of $T_a(T_b)^+$ and $T_b(T_a)^+$ are $U$ and $V$ respectively (after rescaling)*

**Proof:** We can use the above formula for $T_a$ and $T_b$ and compute

$$T_a(T_b)^+ = U D_a D_b^+ U^+$$

Then almost surely over the choice of $a$ and $b$ we have that the diagonals of $D_a D_b^+$ will be distinct – this is where we use the condition that each pair of vectors in $\{w_i\}_i$ is linearly independent.

Hence the above formula for $T_a(T_b)^+$ is an eigendecomposition, and moreover it is unique because its eigenvalues are distinct. We conclude that the eigenvectors of $T_a(T_b)^+$ are indeed the columns of $U$ up to rescaling, and similarly for $V$. ■

Now to complete the proof of the theorem, notice that $u_i$ and $v_i$ as eigenvectors of $T_a(T_b)^+$ and $T_b(T_a)^+$ respectively, have eigenvalues of $(D_a)_{i,i}(D_b)^{-1}_{i,i}$ and $(D_b)_{i,i}(D_a)^{-1}_{i,i}$. (Again, the diagonals of $D_a(D_b)^+$ are distinct almost surely and so $v_i$ is the only eigenvector that $u_i$ could be paired with). Since we only have the factors $u_i \times v_i$ up to scaling, we will need to push the rescaling factor in with $w_i$. Nevertheless we just need to prove that linear system over the $w_i$'s does not have more than one solution (it certainly has one).

**Definition 3.1.6** *The* Khatri-Rao product $\otimes_{KR}$ *between two matrices $U$ and $V$ with the same number of columns is*

$$\Big(U \otimes_{KR} V\Big)_i = u_i \otimes v_i$$

That is the Khatri-Rao product of $U$ and $V$ of size $m \times r$ and $n \times r$ is an $mn \times r$ matrix whose $i$th column is the tensor product of the $i$th column of $U$ and the $i$th column of $V$. The following lemma we leave as an exercise to the reader:

**Lemma 3.1.7** *If $U$ and $V$ are size $m \times r$ and $n \times r$ and have full column rank and $r \leq m + n - 1$ then $U \otimes_{KR} V$ has full column rank too.*

This immediately implies that the linear system over the $w_i$'s has a unique solution. This completes the proof of the theorem.

Note that if $T$ is size $m \times n \times p$ then the conditions of the theorem can only hold if $r \leq \min(m, n)$. There are extensions of the above algorithm that work for higher order tensors even if $r$ is larger than any of the dimensions of its factors [48], [66], [26] and there are interesting applications to overcomplete independent component analysis [66] and learning mixtures of many Gaussians [26], [11].

In the next section, we will show that the above algorithm is stable – in all of the applications in learning we will estimate $T$ from our samples and hence we do not have $T$ exactly.

## 3.2 Perturbation Bounds

In the last section, we gave an algorithm for tensor decomposition when the factors are full-rank, and in this setting its decomposition is unique (up to rescaling).

However in all of our applications we will not be given $T$ exactly but rather we will compute an approximation to it from our samples. Our main goal in this section is to show that even in the presence of noise, the algorithm in Theorem 3.1.3 recovers factors close to the true factors. In later sections, we will simply assume we are given the true tensor $T$ and what we present here is what justifies this simplification.

*This section is somewhat technical, and the reader can feel free to skip it.*

Recall that the main step in Theorem 3.1.3 is to compute an eigendecomposition. Hence our first goal is to establish conditions under which the eigendecomposition itself is stable. More precisely, let $M = UDU^{-1}$, where $D$ is a diagonal matrix. If we are given $\widetilde{M} = M + E$, when can we recover good estimates to $U$?

Intuitively, if any of the diagonal entries in $D$ are close or if $U$ is ill-conditioned, then even a small perturbation $E$ can drastically change the eigendecomposition. We will prove that these are the only things that can go wrong. There will be two main steps. First we need to prove that $\widetilde{M}$ is diagonalizable, and then we can show that the matrix that diagonalizes it must be close to the one that diagonalizes $M$.

## Condition Number

**Definition 3.2.1** *The* condition number *of a matrix $M$ is defined as*

$$\kappa(M) := \frac{\sigma_{\max}(M)}{\sigma_{\min}(M)},$$

*where $\sigma_{\max}(M)$ and $\sigma_{\min}(M)$ are the maximum and minimum singular values of $M$, respectively.*

Consider the basic problem of solving for $x$ in $Mx = b$. Suppose that we are given $M$ exactly, but we only know an estimate $\widetilde{b} = b + e$ of $b$. Here $e$ is an error term. By solving the equation $Mx = b$ using $\widetilde{b}$ instead of $b$, we obtain an estimate $\widetilde{x}$ for $x$. How close is $\widetilde{x}$ to $x$?

We have $\widetilde{x} = M^{-1}\widetilde{b} = x + M^{-1}e = x + M^{-1}(\widetilde{b} - b)$. So

$$\|x - \widetilde{x}\| \leq \frac{1}{\sigma_{\min}(M)} \|b - \widetilde{b}\|.$$

Since $Mx = b$, we also have $\|b\| \leq \sigma_{\max}(M)\|x\|$. It follows that

$$\frac{\|x - \widetilde{x}\|}{\|x\|} \leq \frac{\sigma_{\max}(M)}{\sigma_{\min}(M)} \frac{\|b - \widetilde{b}\|}{\|b\|} = \kappa(M) \frac{\|b - \widetilde{b}\|}{\|b\|}.$$

In other words, the condition number controls the *relative* error when solving a linear system.

## Gershgorin's Disk Theorem

Recall our first intermediate goal is to show that $M + E$ is diagonalizable, and we will invoke the following theorem:

**Theorem 3.2.2** *The eigenvalues of a matrix $M$ are all contained in the following union of disks in the complex plane:*

$$\bigcup_{i=1}^{n} D(M_{ii}, R_i)$$

*where $D(a, b) := \{x \mid \|x - a\| \leq b\} \subseteq \mathbb{C}$ and $R_i = \sum_{j \neq i} |M_{ij}|$.*

**Proof:** Let $(x, \lambda)$ be an eigenvector-eigenvalue pair (note that this is valid even when $M$ is not diagonalizable). Let $i$ denote the coordinate of $x$ with the maximum absolute value. Then $Mx = \lambda x$ gives $\sum_j M_{ij}x_j = \lambda x_i$. So $\sum_{j \neq i} M_{ij}x_j = \lambda x_i - M_{ii}x_i$. We conclude:

$$|\lambda - M_{ii}| = \left|\sum_{j \neq i} M_{ij}\frac{x_j}{x_i}\right| \leq \sum_{j \leq i} |M_{ij}| = R_i.$$

Thus $\lambda \in D(M_{ii}, R_i)$. ■

## Part 1

Now let us return to the question posed at the beginning of the previous section: is $\widetilde{M}$ diagonalizable? Consider

$$U^{-1}\widetilde{M}U = D + U^{-1}EU.$$

The proof that $\widetilde{M}$ is diagonalizable proceeds as follows:

**Part (a)** Since $\widetilde{M}$ and $U^{-1}\widetilde{M}U$ are similar matrices, they have the same set of eigenvalues.

**Part (b)** Moreover we can apply Theorem 3.2.2 to $U^{-1}\widetilde{M}U = D + U^{-1}EU$ and if $U$ is well-conditioned and $E$ is sufficiently small, the radii will be much smaller than the closest pair of diagonal entries in $D$. Hence we conclude that the eigenvalues of $U^{-1}\widetilde{M}U$ and also those of $\widetilde{M}$ are distinct, and hence the latter can be diagonalized.

Thanks to Santosh Vempala for pointing out an error in the original analysis; see also [66] for a more detailed proof along these lines.

## Part 2

Let $\widetilde{M} = \widetilde{U}\widetilde{D}\widetilde{U}^{-1}$. Now we can turn to our second intermediate goal, namely how does this compare to the actual diagonalization $M = UDU^{-1}$?

More specifically, if $(\widetilde{u}_i, \widetilde{\lambda}_i)$ and $(u_i, \lambda_i)$ are corresponding eigenvector-eigenvalue pairs for $\widetilde{M}$ and $M$ respectively, how close is $(\widetilde{u}_i, \widetilde{\lambda}_i)$ to $(u_i, \lambda_i)$? Using the argument in **Part 1** we know that $\widetilde{\lambda}_i \approx \lambda_i$ for each $i$. Furthermore, we assume that when $i \neq j$, the eigenvalues of $M$ have sufficient separation. It remains to check that $\widetilde{u}_i \approx u_i$. Let

$$\sum_j c_j u_j = \widetilde{u}_i.$$

Recall that $\widetilde{M} = M + E$. Left-multiplying both sides of the equation above by $\widetilde{M}$, we get

$$\sum_j c_j \lambda_j u_j + E\widetilde{u}_i = \widetilde{\lambda}_i \widetilde{u}_i. \Longrightarrow \sum_j c_j(\lambda_j - \widetilde{\lambda}_i) u_j = -E\widetilde{u}_i.$$

Let $w_j^T$ be the $j$th row of $U^{-1}$. Left-multiplying both sides of the above equation by $w_j^T$, we get

$$c_j(\lambda_j - \widetilde{\lambda}_i) = -w_j^T E \widetilde{u}_i.$$

Recall we have assumed that the eigenvalues of $M$ are separated. Hence if $E$ is sufficiently small we have that $\lambda_j - \widetilde{\lambda}_i$ is bounded away from zero. Then we can bound the $c_j$'s and this implies that $\widetilde{u}_i$ and $u_i$ are close.

We can qualitatively restate this as follows: Let $\delta$ be the separation between the closest pair of eigenvalues of $M$ and let $\kappa$ be the condition number of $U$. Then if $\|E\| \leq \text{poly}(1/n, 1/\kappa, \delta)$ the norm of the error satisfies $\|\widetilde{U} - U\| \leq \text{poly}(1/n, 1/\kappa, \delta, \|E\|)$.

## Back to Tensor Decompositions

We will introduce some notation to explain the application to tensor decompositions. Let "$\rightarrow$" signify that one matrix converges to another at an inverse polynomial rate (as a function of the number of samples). For example, $\widetilde{T} \rightarrow T$ when $\widetilde{T}$ represents the empirical moments of a distribution (with bounded moments) and $T$ represents its true moments. Also $\widetilde{T}_a = \widetilde{T}(*, *, a) \rightarrow T_a$ and similarly for $b$.

We leave it as an exercise to the reader to check that $\widetilde{T}_b^+ \rightarrow T_b^+$ under natural conditions. It follows that $\widetilde{T}_a\widetilde{T}_b^+ \rightarrow T_a T_b^+$. We have already established that if $E \rightarrow 0$, then the eigendecompositions of $M$ and $M + E$ converge. Finally we conclude that the algorithm in Theorem 3.1.3 computes factors $\widetilde{U}$ and $\widetilde{V}$ which converge to the true factors $U$ and $V$ at an inverse polynomial rate, and a similar proof works for $\widetilde{W}$ and $W$ as well.

### Open Problem: Kruskal rank

We conclude this section with an open problem.

**Definition 3.2.3** *The* Kruskal rank *of a set of vectors $\{u_i\}_i$ is the maximum $r$ such that all subset of $r$ vectors are linearly independent.*

We will see later that it is $NP$-hard to compute the Kruskal rank. Nevertheless, there are strong uniqueness theorems for tensor decompositions (based on this parameter) for which there is no known algorithmic proof:

**Theorem 3.2.4 (Kruskal)** *Let $T = \sum_{i=1}^{r} u_i \otimes v_i \otimes w_i$ and let $k_u, k_v$ and $k_w$ be the Kruskal ranks of $\{u_i\}_i$, $\{v_i\}_i$, and $\{w_i\}_i$ respectively. If $k_u + k_v + k_w \geq 2r + 2$ then $T$ has rank $r$ and this decomposition of $T$ is unique up to rescaling.*

**Open Question 1** *Is there an efficient algorithm for tensor decompositions under* any *natural conditions, for $r = (1 + \varepsilon)n$ for any $\varepsilon > 0$?*

For example, it is natural to consider a smoothed analysis model for tensor decomposition [26] where the factors of $T$ are perturbed and hence not adversarially chosen. The above uniqueness theorem would apply up to $r = 3/2n - O(1)$ but there are no known algorithms for tensor decomposition in this model for $r = (1 + \epsilon)n$ (although there are much better algorithms for higher-order tensors).

## 3.3 Phylogenetic Trees and HMMs

Here we describe an application of tensor decompositions to phylogenetic reconstruction and HMMs.

### The Model

A phylogenetic model has the following components:

(a) A rooted binary tree with root $r$, where the leaves do not necessarily have the same depth.

The biological interpretation is that the leaves represent *extant* species (ones that are still living), and the internal nodes represent speciation events.

(b) A set $\Sigma$ of states, for example $\Sigma = \{A, C, G, T\}$. Let $k = |\Sigma|$.

(c) A Markov model on the tree; i.e. a distribution $\pi_r$ on the state of the root and a transition $P^{uv}$ matrix for each edge $(u, v)$.

We can generate a sample from the model as follows: We choose a state for the root according to $\pi_r$ and for each node $v$ with parent $u$ we choose the state of $v$ according to the distribution defined by the $i$th row of $P^{uv}$, where $i$ is the state of $u$. Alternatively, we can think of $s(\cdot) : V \to \Sigma$ as a random function that assigns states to vertices where the marginal distribution on $s(r)$ is $\pi_r$ and

$$P^{uv}_{ij} = \mathbb{P}(s(v) = j | s(u) = i),$$

Note that $s(v)$ is independent of $s(t)$ conditioned on $s(u)$ whenever the (unique) shortest path from $v$ to $t$ in the tree passes through $u$.

Our goal is to learn the above model - both the tree and the transition matrices - from a polynomial number of random samples. We will assume that the transition matrices are full rank, in which case it is easy to see that we could root the tree arbitrarily. To connect this back with biology, here we are assuming we have sequenced each of the extant species and that moreover these sequences have already been properly aligned. We think of the $i$th symbol in each of these sequences as being an independent sample from the above model, and we want to reconstruct the evolutionary tree that led to the species we have today as well as get some understanding of how long these evolutionary branches were. We mention as a caveat that one of the most interesting and challenging problems in computational biology is to perform multiple sequence alignment, and here we have assumed that a fully aligned set of sequences is our starting point. Moreover our model for evolution is simplistic in that we only only point mutations instead of insertions, deletions and cross-over.

This is really two separate learning goals: Our approach for finding the topology will follow the foundational work of Steel [109] and Erdos, Steel, Szekely, and Warnow [57]. And from this, we can apply tensor methods to find the transition matrices following the approach of Chang [36] and later Mossel and Roch [96].

## Finding the Topology

The basic idea here is to define an appropriate distance function [109] on the edges of the tree, so that we can approximately compute the distance between leaves from our samples and then construct the tree.

### Defining a Tree Metric

Suppose first that, for leaves $a$ and $b$, we have access to the true values of $F^{ab}$, where

$$F^{ab}_{ij} = \mathbb{P}(s(a) = i, s(b) = j).$$

In [109], Steel defined a distance metric on the tree in a way that allows us to compute the distances between leaves $a$ and $b$, given $F^{ab}$. In particular, let

$$\psi_{ab} := -\ln|\det(F^{ab})|.$$

Steel showed that

$$\psi_{ab} = \sum_{(u,v)\in p_{ab}} \nu_{uv},$$

where $p_{ab}$ is the unique path in the tree from $a$ to $b$, and

$$\nu_{uv} = -\ln|\det(P^{uv})| + \frac{1}{2}\ln\left(\prod_{i\in[k]} \pi_u(i)\right) - \frac{1}{2}\ln\left(\prod_{i\in[k]} \pi_v(i)\right).$$

He then showed that $\nu_{uv}$ is always non-negative (which is not obvious), and hence $\psi$ is indeed a metric.

The important point is that we can estimate $F^{ab}$ from our samples, and hence we can (approximately) compute $\psi_{ab}$ on the leaves.

### Reconstructing Quartets

Here we will use $\psi$ to compute the topology. Fix four leaves $a$, $b$, $c$, and $d$, and there are exactly three possible induced topologies between these leaves, given in Figure 3.1. (Here by induced topology, we mean delete edges not on any shortest path between any pair of the four leaves, and contract paths to a single edge if possible). Our goal is to determine which of these induced topologies is the true topology, given the pairwise distances. Consider topology (a) on the left of Figure 3.1; in this case, we have

$$\psi(a,b) + \psi(c,d) < \min\{\psi(a,c) + \psi(b,c), \psi(a,d) + \psi(b,d)\},$$

Thus we can determine which is the true induced topology by computing three values $\psi(a,b) + \psi(c,d)$, $\psi(a,c) + \psi(b,c)$, and $\psi(a,d) + \psi(b,d)$. Whichever is the smallest determines the induced topology because whichever nodes are paired up are the ones with a common parent (again in the induced topology).

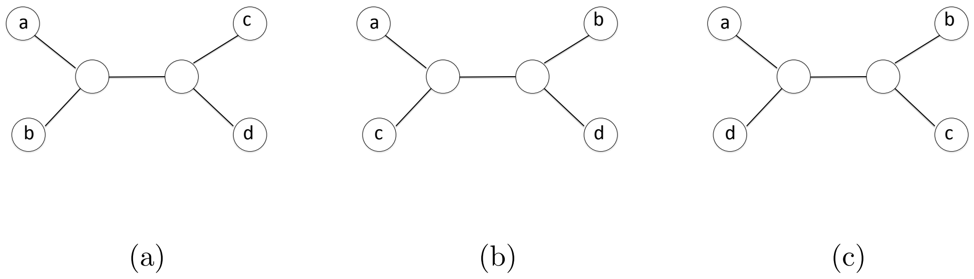

Figure 3.1: Possible quartet topologies

Indeed from just these quartet tests we can recover the topology of the tree. For example, a pair of leaves $a, b$ have the same parent if and only if these nodes always have a common parent in the induced topology for each quartet test. Hence we can pair up all of the leaves so that they have the same parent, and it is not hard to extend this approach to recover the topology of the tree.

### Handling Noise

Note that we can only approximate $F^{ab}$ from our samples. This translates into a good approximation of $\psi_{ab}$ when $a$ and $b$ are close, but is noisy when $a$ and $b$ are far away. The approach in [57] of Erdos, Steel, Szekely, and Warnow is to only use quartets where all of the distances are short.

## Finding the Transition Matrices

Here we will assume that we know the topology of the tree and $T^{abc}$ for all triplets $a, b, c$ of leaves, where

$$T^{abc}_{ijk} = \mathbb{P}(s(a) = i, s(b) = j, s(c) = k).$$

(which we can approximate from random samples). Then consider the unique node that lies on all of the shortest paths among $a$, $b$, and $c$; since we can reroot the tree arbitrarily let this node be $r$. Then

$$\begin{aligned} T^{abc} &= \sum_{\ell} \mathbb{P}(s(r) = \ell)\mathbb{P}(s(a) = \cdot | s(r) = \ell) \otimes \mathbb{P}(s(b) = \cdot | s(r) = \ell) \otimes \mathbb{P}(s(c) = \cdot | s(r) = \ell) \\ &= \sum_{\ell} \mathbb{P}(s(r) = \ell) P^{ra}_{\ell} \otimes P^{rb}_{\ell} \otimes P^{rc}_{\ell} \end{aligned}$$

where we have used $P^{rx}_{\ell}$ to denote the $\ell$th row of the transition matrix $P^{rx}$.

We can now apply the algorithm in Section 3.1 to compute a tensor decomposition of $T$ whose factors are unique up to rescaling. Furthermore the factors are probability distributions and hence we can compute their proper normalization. We will call this procedure a *star test*. (Indeed, the algorithm for tensor decompositions in Section 3.1 has been re-discovered many times and it is also called Chang's lemma [36]).

In [96], Mossel and Roch use this approach to find the transition matrices of a phylogenetic tree, given the tree topology, as follows. Let us assume that $u$ and $v$ are internal nodes and that $w$ is a leaf. Furthermore suppose that $v$ lies on the shortest path between $u$ and $w$. The basic idea is to write

$$P^{uw} = P^{uv} P^{vw}$$

and if we can find $P^{uw}$ and $P^{vw}$ (using the star tests above) then we can compute $P^{uv} = P^{uw}(P^{vw})^{-1}$ since we have assumed that the transition matrices are invertible.

However there are two serious complications:

(a) As in the case of finding the topology, long paths are very noisy.

Mossel and Roch showed that one can recover the transition matrices also using only queries to short paths.

(b) We can only recover the tensor decomposition up to relabeling.

In the above star test, we could apply any permutation to the states of $r$ and permute the rows of the transition matrices $P^{ra}$, $P^{rb}$ and $P^{rc}$ accordingly so that the resulting joint distribution on $a, b$ and $c$ is unchanged.

However the approach of Mossel and Roch is to work instead in the framework of PAC learning [114] where the goal is to learn a generative model that produces almost the same joint distribution on the leaves. (In particular, if there are multiple ways to label the internal nodes to produce the same joint distribution on the leaves, we are indifferent to them).

**Remark 3.3.1** *HMMs are a special case of phylogenetic trees where the underlying topology is a caterpillar. But note that for the above algorithm, we need that the transition matrices* and *the observation matrices are full-rank.*

More precisely, we require that the transition matrices are invertible and that the observation matrices whose row space correspond to a hidden node and whose column space correspond to the output symbols each have full row rank.

## Beyond Full Rank?

The algorithm above assumed that all transition matrices are full rank. In fact if we remove this assumption, then it is easy to embed an instance of the *noisy parity problem* [31] which is a classic hard learning problem. Let us first define this problem *without noise*:

Let $S \subset [n]$, and choose $X_j \in \{0,1\}^n$ independently and uniformly at random, for $j = 1, \ldots, m$. Given $X_j$ and $b_j = \chi_S(X_j) := \sum_{i \in S} X_j(i) \mod 2$ for each $j$, the goal is to recover $S$.

This is quite easy: Let $A$ be the matrix whose $j$th row is $X_j$ and let $b$ be a column vector whose $j$th entry is $b_j$. It is straightforward to see that $\mathbf{1}_S$ is a solution to the linear system $Ax = b$ where $\mathbf{1}_S$ is the indicator function for $S$. Furthermore if we choose $\Omega(n \log n)$ samples then $A$ is w.h.p. full column rank and so this solution is unique. We can then find $S$ by solving a linear system over $GF(2)$.

Yet a slight change in the above problem does not change the sample complexity but makes the problem drastically harder. The noisy parity problem is the same as above but for each $j$ we are independently given the value $b_j = \chi_S(X_j)$ with probably 2/3 and otherwise $b_j = 1 - \chi_S(X_j)$. The challenge is that we do not know which labels have been flipped.

**Claim 3.3.2** *There is a brute-force algorithm that solves the noisy parity problem using $O(n \log n)$ samples*

**Proof:** For each $T$, calculate $\chi_T(X_j)b_j$ over the samples. Indeed $\chi_T(X_j)$ and $b_j$ are correlated if and only if $S = T$. ■

This algorithm runs in time $2^n$ (roughly). The state-of-the-art due to Blum, Kalai, and Wasserman [31] has running time and sample complexity $2^{n/\log n}$. It is widely believed that there is no polynomial time algorithm for noisy parity even given any polynomial number of samples. *This is an excellent example of a problem whose sample complexity and computational complexity are (conjectured) to be wildly different.*

Next we show how to embed samples from a noisy parity problem into an HMM, however to do so we will make use of transition matrices that are not full rank. Consider an HMM that has $n$ hidden nodes, where the $i$th hidden node encodes is used to represent the $i$th coordinate of $X$ and the running parity

$$\chi_{S_i}(X) := \sum_{i' \leq i, i' \in S} X(i') \mod 2.$$

Hence each node has four possible states. We can define the following transition matrices. Let $s(i) = (x_i, s_i)$ be the state of the $i$th internal node where $s_i = \chi_{S_i}(X)$.

We can define the following transition matrices:

$$\text{if } i+1 \in S \qquad P^{i+1,i} = \begin{cases} \frac{1}{2} & (0, s_i) \\ \frac{1}{2} & (1, s_i + 1 \mod 2) \\ 0 & \text{otherwise} \end{cases}$$

$$\text{if } i+1 \notin S \qquad P^{i+1,i} = \begin{cases} \frac{1}{2} & (0, s_i) \\ \frac{1}{2} & (1, s_i) \\ 0 & \text{otherwise} \end{cases} .$$

At each internal node we observe $x_i$ and at the last node we also observe $\chi_S(X)$ with probability $2/3$ and otherwise $1 - \chi_S(X)$. Each sample from the noisy parity problem is a set of observations from this HMM, and if we could learn the transition matrices of it we would necessarily learn $S$ and solve the noisy parity problem.

Note that here the observation matrices are certainly not full rank because we only observe two possible emissions even though each internal node has four possible states! Hence these problems become much harder when the transition (or observation) matrices are not full rank!

## 3.4 Community Detection

Here we give applications of tensor methods to community detection. There are many settings in which we would like to discover *communities* - that is, groups of people with strong ties. Here we will focus on graph theoretic approaches, where we will think of a community as a set of nodes that are better connected to each other than to nodes outside of the set. There are many ways we could formalize this notion, each of which would lead to a different optimization problem e.g. *sparsest cut* or *k-densest subgaph*.

However each of these optimization problems is $NP$-hard, and even worse are hard to approximate. Instead, we will formulate our problem in an average-case model where there is an underlying community structure that is used to generate a random graph, and our goal is to recover the true communities from the graph with high probability.

## Block Stochastic Model

Here we introduce the block stochastic model, which is used to generate a random graph on $V$ with $|V| = n$. Additionally, the model is specified by parameters $p$ and $q$ and a partitioning specified by a function $\pi$:

- $\pi : V \to [k]$ partitions the vertices $V$ into $k$ *disjoint* groups (we will relax this condition later);
- Each possible edge $(u, v)$ is chosen *independently* with:

$$\Pr[(u, v) \in E] = \begin{cases} q & \pi(u) = \pi(v) \\ p & \text{otherwise} \end{cases}.$$

In our setting we will set $q > p$, but this model has been studied in cases where $q < p$ too. (In particular, when $q = 0$ we could ask to find a $k$-coloring of this random graph). Regardless, we observe a random graph generated from the above model and our goal is to recover the partition described by $\pi$.

When is this *information theoretically* possible? In fact even for $k = 2$ where $\pi$ is a bisection, we need

$$q - p > \Omega\Big(\sqrt{\frac{\log n}{n}}\Big)$$

in order for the true bisection to be the uniquely smallest cut that bisects the random graph $G$ with high probability. If $q - p$ is smaller, then it is not even information theoretically possible to find $\pi$. Indeed, we should also require that each part of the partition is large, and for simplicity we will assume that $k = O(1)$ and $|\{u|\pi(u) = i\}| = \Omega(n)$.

There has been a long line of work on partitioning random graphs in the block stochastic model, culminating in the work of McSherry [91]:

**Theorem 3.4.1** *[91] There is an efficient algorithm that recovers $\pi$ (up to relabeling) if*

$$\frac{q - p}{q} > c\sqrt{\frac{\log n/\delta}{qn}}$$

*and succeeds with probability at least $1 - \delta$.*

This algorithm is based on spectral clustering, where we think of the observed adjacency matrix as the sum of a rank $k$ matrix which encodes $\pi$ and an error term. If the error is small, then we can recover something close to the true rank $k$ matrix by finding the best rank $k$ approximation to the adjacency matrix. For the full details, see [91].

We will instead follow the approach in Anandkumar et al [9] that makes use of tensor decompositions instead. In fact, the algorithm of [9] also works in the *mixed membership* model where we allow each node to be a distribution over $[k]$. Then if $\pi^u$ and $\pi^v$ are the probability distributions for $u$ and $v$, the probability of an edge $(u, v)$ is $\sum_i \pi_i^u \pi_i^v q + \sum_{i \neq j} \pi_i^u \pi_j^v p$. We can interpret this probability as: $u$ and $v$ choose a community according to $\pi^u$ and $\pi^v$ respectively, and if they choose the same community there is an edge with probability $q$ and otherwise there is an edge with probability $p$.

Recall that in order to apply tensor decomposition methods what we really need are conditionally independent random variables! In fact we will get such random variables based on counting three stars.

## Counting Three Stars

We will partition $V$ into four sets (arbitrarily) $X$, $A$, $B$, and $C$. Let $\Pi \in \{0,1\}^{V \times k}$ represent the (unknown) assignment of vertices to communities, such that each row of $\Pi$ contains exactly one 1. Also let $R \in \mathbb{R}^{k \times k}$ be the matrix describing the probability of each pair of communities having an edge. In particular,

$$(R)_{ij} = \begin{cases} q & i = j \\ p & i \neq j \end{cases}.$$

Consider the product $\Pi R$. The $i$th column of $\Pi R$ encodes the probability that an edge occurs from a vertex in community $i$ to a given other vertex:

$$(\Pi R)_{xi} = \Pr[(x, a) \in E | \pi(a) = i].$$

We will use $(\Pi R)_i^A$ to denote the matrix $\Pi R$ restricted to the $i$th column and the rows in $A$, and similarly for $B$ and $C$. Moreover let $p_i$ be the fraction of nodes in $X$ that are in community $i$. Then consider the following tensor

$$T := \sum_i p_i T_x = \sum_i p_i (\Pi R)_i^A \otimes (\Pi R)_i^B \otimes (\Pi R)_i^C.$$

The key claim is:

**Claim 3.4.2** *Let $a \in A$, $b \in B$ and $c \in C$; then $T_{a,b,c}$ is exactly the probability that a random node $x \in X$ is connected to $a$, $b$ and $c$.*

This is immediate from the definitions above. In particular if we look at whether $(x, a)$, $(x, b)$ and $(x, c)$ are edges in $G$, these are conditionally independent random variables. Then we need to prove:

(a) If $|X| = \widetilde{\Omega}(|A||B||C|/\epsilon^2)$, then we can estimate $T$ accurately

(b) The factors $\{(\Pi R)^A_i\}_i$, $\{(\Pi R)^B_i\}_i$, and $\{(\Pi R)^B_i\}_i$ are linearly independent, and hence the tensor decomposition of $T$ is unique by Theorem 3.1.3

More precisely, we need these factors to be well-conditioned so that we can approximate them from an approximation $\widetilde{T}$ to $T$. See Section 3.2.

(c) We can recover $\pi$ from $\{(\Pi R)^A_i\}_i$ up to relabeling.

**Part (a)** Let $\{X_{a,b,c}\}_{a,b,c}$ be a partition of $X$ into almost equal sized sets, one for each $a \in A$, $b \in B$ and $c \in C$. Then

$$\widetilde{T}_{a,b,c} = \frac{|\{x \in X_{a,b,c} | (x, a), (x, b), (x, c) \in E\}|}{|X_{a,b,c}|}$$

will be close to $T_{a,b,c}$ with high probability. We can then use a union bound.

**Part (b)** It is easy to see that $R$ is full rank and moreover if we choose $A$, $B$ and $C$ at random then if each community is large enough, with high probability each community will be well-represented in $A$, $B$ and $C$ and hence the factors $\{(\Pi R)^A_i\}_i$, $\{(\Pi R)^B_i\}_i$, and $\{(\Pi R)^B_i\}_i$ will be non-negligibly far from linearly dependent.

**Part (c)** Note that if we have a good approximation to $\{(\Pi R)^A_i\}_i$ then we can partition $A$ into communities. In turn, if $A$ is large enough then we can extend this partitioning to the whole graph: We add a node $x \notin A$ to community $i$ if and only if the fraction of nodes $a \in A$ with $\pi(a) = i$ that $x$ is connected to is close to $q$. With high probability, this will recover the true communities.

However for a full proof of the algorithm see [9]. Anandkumar et al also give an algorithm for mixed membership models where each $\pi_u$ is chosen from a Dirichlet. We will not cover this latter extension because we will instead explain those types of techniques in the setting of topic models next.

We note that there are powerful extensions to the block-stochastic model that are called *semi-random models*. Roughly, these models allow an "adversary" to add edges between nodes in the same cluster and delete edges between clusters after $G$

is generated. If $\pi$ is the best partitioning of $G$, then this is only more true after the changes. Interestingly, many spectral algorithms breakdown in this more flexible model, but there are elegant techniques for recovering $\pi$ even in this more general setting (see [60], [59]).

## 3.5 Extensions to Mixed Models

Here we will extend tensor spectral models to work with (some) mixed models.

### Pure Topic Model

First we describe an easy application of tensor methods to pure topic models (see [10]). Recall that there is an unknown topic matrix $A$ and we obtain samples from the following model:

(a) Choose topic $i$ for document $j$ with probability $p_i$

(b) Choose $N_j$ words according to the distribution $A_i$

If each document has at least three words, we can define the tensor $\widetilde{T}$ where $\widetilde{T}_{a,b,c}$ counts the fraction of documents in our sample whose first word, second word and third word are $a$, $b$ and $c$ respectively. Then it is easy to see that if the number of documents is large enough then $\widetilde{T}$ converges to:

$$T = \sum_{i=1}^{r} p_i A_i \otimes A_i \otimes A_i$$

In order to apply the algorithm in Section 3.1, we just need that $A$ has full column rank. In this case the factors in the decomposition are unique up to rescaling, and the algorithm will find them. Finally, each column in $A$ is a distribution and so we can properly normalize these columns and compute the values $p_i$ too. Recall in Section 3.2 we analyzed the noise tolerance of our tensor decomposition algorithm. It is easy to see that this algorithm recovers a topic matrix $\widetilde{A}$ and a distribution $\{\widetilde{p}_i\}_i$ that is $\epsilon$-close to $A$ and $\{p_i\}_i$ respectively with high probability if we are given at least poly$(n, 1/\epsilon, 1/\sigma_r)$ documents of length at least three, where $n$ is the size of the vocabulary and $\sigma_r$ is the smallest singular value of $A$.

We will refer to this as an application of tensor methods to *pure* models, since each document is described by one and only one topic. Similarly, in our application to community detection, each node belonged to one and only one community.

Finally, in our application to phylogenetic reconstruction, each hidden node was in one and only one state. Note however that in the context of topic models, it is much more realistic to assume that each document is itself a mixture of topics and we will refer to these as *mixed* models.

## Latent Dirichlet Allocation

Here we will give a tensor spectral algorithm for learning a very popular mixed model, called Latent Dirichlet Allocation [30]. Let $\Delta := \{x \in \mathbb{R}^r : x \geq 0, \sum_i x_i = 1\}$ denotes the $r$-dimensional simplex. Then we obtain samples from the following model:

(a) Choose a mixture over topics $w_j \in \Delta$ for document $j$ according to the Dirichlet distribution $\mathrm{Dir}(\{\alpha_i\}_i)$

(b) Repeat $N_j$ times: choose a topic $i$ from $w_j$, and choose a word according to the distribution $A_i$.

The Dirichlet distribution is defined as

$$p(x) \propto \prod_i x_i^{\alpha_i - 1} \text{ for } x \in \Delta$$

Note that if documents are long (say $N_j > n \log n$) then in a pure topic model, pairs of documents often have nearly identical empirical distributions on words. But this is no longer the case in mixed models like the one above.

The basic issue in extending our tensor spectral approach to mixed models is that the tensor $\widetilde{T}$ that counts triples of words converges to

$$T = \sum_{ijk} D_{ijk} A_i \otimes A_j \otimes A_k$$

where $D_{i,j,k}$ is the probability that the first three words in a random document are generated from topics $i$, $j$ and $k$ respectively. In a pure topic model, $D_{i,j,k}$ was diagonal but for a mixed model it is not!

**Definition 3.5.1** *A Tucker decomposition of $T$ is*

$$T = \sum_{i,j,k} D_{i,j,k} a_i \otimes b_j \otimes c_k$$

*where $D$ is $r_1 \times r_2 \times r_3$. We call $D$ the core tensor.*

This is different than the standard definition for a tensor decomposition where we only summed over $i = j = k$. The good news is that computing a Tucker decomposition of a tensor is easy. Indeed we can always set $r_1$ equal to the dimension of $\text{span}(\{T_{i,*,*}\}_i)$, and similarly for $r_2$ and $r_3$. However the bad news is that a Tucker decomposition is in general not unique, so even if we are given $T$ we cannot necessarily compute the above decomposition whose factors are the topics in the topic model.

How can we extend the tensor spectral approach to work with mixed models? The elegant approach of Anandkumar et al [8] is based on the following idea:

**Lemma 3.5.2**

$$\begin{aligned} T &= \sum_{ijk} D_{ijk} A_i \otimes A_j \otimes A_k \\ S &= \sum_{ijk} \widetilde{D}_{ijk} A_i \otimes A_j \otimes A_k \\ \implies T - S &= \sum_{ijk} (D_{ijk} - \widetilde{D}_{ijk}) A_i \otimes A_j \otimes A_k \end{aligned}$$

**Proof:** The proof is a simple exercise in multilinear algebra. ■

Hence if we have access to other tensors $S$ which can be written using the same factors $\{A_i\}_i$ in its Tucker decomposition, we can subtract $T$ and $S$ and hope to make the core tensor diagonal. We can think of $D$ as being the third order moments of a Dirichlet distribution in our setting. What other tensors do we have access to?

## Other Tensors

We described the tensor $T$ based on the following experiment: Let $T_{a,b,c}$ be the probability that the first three words in a random document are $a$, $b$ and $c$ respectively. But we could just as well consider alternative experiments. The three experiments we will need in order to given a tensor spectral algorithm for LDA are:

(a) Choose three documents at random, and look at the first word of each document

(b) Choose two documents at random, and look at the first two words of the first document and the first word of the second document

(c) Choose a document at random, and look at its first three words

These experiments result in tensors whose factors are the same, but whose cores differ in their natural Tucker decomposition.

**Definition 3.5.3** *Let $\mu$, $M$ and $D$ be the first, second and third order moments of the Dirichlet distribution.*

More precisely, let $\mu_i$ be the probability that the first word in a random document was generated from topic $i$. Let $M_{i,j}$ be the probability that the first and second words in a random document are generated from topics $i$ and $j$ respectively. And as before, let $D_{i,j,k}$ be the probability that the first three words in a random document are generated from topics $i$, $j$ and $k$ respectively. Then let$T^1$, $T^2$ and $T^3$ be the expectation of the first, second and third experiments respectively.

**Lemma 3.5.4** *(a)* $T^1 = \sum_{i,j,k} [\mu \otimes \mu \otimes \mu]_{i,j,k} A_i \otimes A_j \otimes A_k$

*(b)* $T^2 = \sum_{i,j,k} [M \otimes \mu]_{i,j,k} A_i \otimes A_j \otimes A_k$

*(c)* $T^3 = \sum_{i,j,k} D_{i,j,k} A_i \otimes A_j \otimes A_k$

**Proof:** Let $w_1$ denote the first word and let $t_1$ denote the topic of $w_1$ (and similarly for the other words). We can expand $\mathbb{P}[w_1 = a, w_2 = b, w_3 = c]$ as:

$$\sum_{i,j,k} \mathbb{P}[w_1 = a, w_2 = b, w_3 = c | t_1 = i, t_2 = j, t_3 = k] \mathbb{P}[t_1 = i, t_2 = j, t_3 = k]$$

and the lemma is now immediate. ■

Note that $T^2_{a,b,c} \neq T^2_{a,c,b}$ because two of the words come from the same document. Nevertheless, we can symmetrize $T^2$ in the natural way: Set $S^2_{a,b,c} = T^2_{a,b,c} + T^2_{b,c,a} + T^2_{c,a,b}$. Hence $S^2_{a,b,c} = S^2_{\pi(a),\pi(b),\pi(c)}$ for any permutation $\pi : \{a, b, c\} \to \{a, b, c\}$.

Our main goal is to prove the following identity:

$$\alpha_0^2 D + 2(\alpha_0 + 1)(\alpha_0 + 2)\mu^{\otimes 3} - \alpha_0(\alpha_0 + 2) M \otimes \mu(\text{all three ways}) = \text{diag}(\{p_i\}_i)$$

where $\alpha_0 = \sum_i \alpha_i$. Hence we have that

$$\alpha_0^2 T^3 + 2(\alpha_0 + 1)(\alpha_0 + 2) T^1 - \alpha_0(\alpha_0 + 2) S^2 = \sum_i p_i A_i \otimes A_i \otimes A_i$$

The important point is that we can estimate the terms on the left hand side from our sample (if we assume we know $\alpha_0$) and we can apply the algorithm from Section 3.1 to the tensor on the right hand side to recover the topic model, provided that $A$ has full column rank. In fact, we can compute $\alpha_0$ from our samples (see [8]) but we will focus instead on proving the above identity.

## Moments of the Dirichlet

The main identity that we would like to establish is just a statement about the moments of a Dirichlet distribution. In fact, we can think about the Dirichlet as instead being defined by the following combinatorial process:

(a) Initially, there are $\alpha_i$ balls of each color $i$

(b) Repeat $C$ times: choose a ball at random, place it back with one more of its own color

This process gives an alternative characterization of the Dirichlet distribution, from which it is straightforward to calculate:

(a) $\mu = [\frac{\alpha_1}{\alpha_0}, \frac{\alpha_2}{\alpha_0}, ..., \frac{\alpha_r}{\alpha_0}]$

(b) $M_{i,j} = \begin{cases} \frac{\alpha_i(\alpha_i+1)}{\alpha_0(\alpha_0+1)} & i = j \\ \frac{\alpha_i\alpha_j}{\alpha_0(\alpha_0+1)} & \text{otherwise} \end{cases}.$

(c) $T_{i,j,k} = \begin{cases} \frac{\alpha_i(\alpha_i+1)(\alpha_i+2)}{\alpha_0(\alpha_0+1)(\alpha_0+2)} & i = j = k \\ \frac{\alpha_i(\alpha_i+1)\alpha_k}{\alpha_0(\alpha_0+1)(\alpha_0+2)} & i = j \neq k \\ \frac{\alpha_i\alpha_j\alpha_k}{\alpha_0(\alpha_0+1)(\alpha_0+2)} & i, j, k \text{ distinct} \end{cases}.$

For example for $T_{i,i,k}$ this is the probability that the first two balls are color $i$ and the third ball is color $k$. The probably that the first ball is color $i$ is $\frac{\alpha_i}{\alpha_0}$ and since we place it back with one more of its own color, the probability that the second ball is color $i$ as well is $\frac{\alpha_i+1}{\alpha_0+1}$. And the probability that the third ball is color $k$ is $\frac{\alpha_k}{\alpha_0+2}$. It is easy to check the above formulas in the other cases too.

Note that it is much easier to think about only the *numerators* in the above formulas. If we can prove that following relation for just the numerators

$$D + 2\mu^{\otimes 3} - M \otimes \mu(\text{all three ways}) = \text{diag}(\{2\alpha_i\}_i)$$

it is easy to check that we would obtain our desired formula by multiplying through by $\alpha_0^3(\alpha_0 + 1)(\alpha_0 + 2)$.

**Definition 3.5.5** *Let $R = num(D) + num(2\mu^{\otimes 3}) - num(M \otimes \mu)$(all three ways)*

Then the main lemma is:

**Lemma 3.5.6** $R = diag(\{2\alpha_i\}_i)$

We will establish this by a case analysis:

**Claim 3.5.7** *If $i, j, k$ are distinct then $R_{i,j,k} = 0$*

This is immediate since the $i, j, k$ numerator of $D$, $\mu^{\otimes 3}$ and $M \otimes \mu$ are all $\alpha_i \alpha_j \alpha_k$.

**Claim 3.5.8** $R_{i,i,i} = 2\alpha_i$

This is also immediate since the $i, i, i$ numerator of $D$ is $\alpha_i(\alpha_i + 1)(\alpha_i + 2)$ and similarly the numerator of $\mu^{\otimes 3}$ is $\alpha_i^3$. Finally, the $i, i, i$ numerator of $M \otimes \mu$ is $\alpha_i^2(\alpha_i + 1)$. The case that requires some care is:

**Claim 3.5.9** *If $i \neq k$, $R_{i,i,k} = 0$*

The reason this case is tricky is because the terms $M \otimes \mu$(all three ways) do not all count the same. If we think of $\mu$ along the third dimension of the tensor then the $i^{th}$ topic occurs twice in the same document, but if instead we think of $\mu$ as along either the first or second dimension of the tensor, even though the $i^{th}$ topic occurs twice it does not occur twice in the same document. Hence the numerator of $M \otimes \mu$(all three ways) is $\alpha_i(\alpha_i + 1)\alpha_k + 2\alpha_i^2\alpha_k$. Also, the numerator of $D$ is $\alpha_i(\alpha_i + 1)\alpha_k$ and the numerator of $\mu^{\otimes 3}$ is again $\alpha_i^2\alpha_k$.

These three claims together establish the above lemma. Even though the tensor $T^3$ that we could immediately decompose in a pure topic model no longer has a diagonal core tensor in a mixed model, at least in the case of LDA we can still find a formula (each of whose terms we can estimate from our samples) that diagonalizes the core tensor. This yields:

**Theorem 3.5.10** *[8] There is a polynomial time algorithm to learn a topic matrix $\widetilde{A}$ that is $\epsilon$ close to the true $A$ in a Latent Dirichlet Allocation model, provided we are given at least $poly(n, 1/\epsilon, 1/\sigma_r, 1/\alpha_{min})$ documents of length at least thee, where $n$ is the size of the vocabulary and $\sigma_r$ is the smallest singular value of $A$ and $\alpha_{min}$ is the smallest $\alpha_i$.*

Similarly, there are algorithms for community detection in mixed models too, where for each node $u$ we choose a distribution $\pi_u$ over clusters from a Dirichlet distribution [9]. However these algorithms seem to be quite dependent on the assumption that we use a Dirichlet distribution, and it seems hard to generalize these algorithms to any other natural distributions.

## 3.6 Independent Component Analysis

We can think about the tensor methods we have developed as a way to use higher order moments to learn the parameters of a distribution (e.g. for phylogenetic trees, HMMs, LDA, community detection) through tensor decomposition. Here we will give another style of using the method of moments through an application to *independent component analysis* which was introduced by Comon [42].

This problem is simple to define: Suppose we are given samples of the form

$$y = Ax + b$$

where we know that the variables $x_i$ are independent and the linear transformation $(A, b)$ is unknown. The goal is to learn $A, b$ efficiently from a polynomial number of samples. This problem has a long history, and the typical motivation for it is to consider a hypothetical situation called the *cocktail party problem*

> We have $N$ microphones and $N$ conversations going on in an room. Each microphone hears a superposition of the conversations given by the corresponding rows of $A$. If we think of the conversations as independent and memoryless, can we disentangle them?

Such problems are also often referred to as *blind source separation.* We will follow an approach of Frieze, Jerrum and Kannan [62].

### Step 1

We can always transform the problem $y = Ax + b$ into $y = \widehat{A}\widehat{x} + \widehat{b}$ so that $\mathbb{E}[\widehat{x}_i] = 0$ and $\mathbb{E}[\widehat{x}_i^2] = 1$ for all $i$ by setting $\widehat{b} = b + A\,\mathbb{E}[x]$ and $\widehat{A}_i = A_i \mathrm{std}(x_i)$ where $\mathrm{std}(x_i)$ is the standard deviation of $x_i$.

Note that the distribution on $y$ has not changed, but we have put $(A, x)$ into a canonical form since we anyways cannot distinguish between a pair of linear transformations that have the same canonical form. So without loss of generality we have reduced to the problem of learning

$$y = Ax + b$$

where for all $i$, $E[x_i] = 0, \mathbb{E}[x_i^2] = 1$. Also we can set $b = 0$ since we can easily learn $b$. The crucial assumption that we will make is that $A$ is non-singular.

## Step 2

$\mathbb{E}[yy^T] = \mathbb{E}[Axx^TA^T] = AA^T$

The last equality follows from the condition that $E[x_i] = 0, E[x_i^2] = 1$ and each $x_i$ is independent. Hence we have access to $M = AA^T$ which we can learn up to arbitrary precision by taking sufficiently many random samples. But what does this tell us about $A$? We have encountered this problem before: $M$ does not uniquely define $A$, and our approach was to consider higher order tensors. This time we will proceed in a different manner.

Since $M \succ 0$ we can find $B$ such that $M = BB^T$. How are $B$ and $A$ related?

In fact, we can write

$$BB^T = AA^T \Rightarrow B^{-1}AA^T(B^{-1})^T = I$$

and this implies that $B^{-1}A$ is orthogonal since a square matrix times its own transpose is the identity if and only if it is orthogonal. So we have learned $A$ up to an unknown rotation. Can we hope to learn the rotation $R = B^{-1}A$? Hint: what if each $x_i$ is a standard Gaussian?

In this case, $Rx$ is also a spherical Gaussian and hence we cannot hope to learn $R$ without an additional assumption. In fact, this is the only case that can go wrong: Provided the $x_i$'s are not Gaussian, we will be able to learn $R$ and hence $A$. For simplicity let us assume that each $x_i$ is $\pm 1$ hence $\mathbb{E}[x_i^4] = 1$ and yet the fourth moment of a Gaussian is three. Note that we can apply $B^{-1}$ to our samples and hence we can imagine that we are getting samples from $y = Rx$. The key to our analysis is the following functional

$$F(u) := \mathbb{E}[(u^TRx)^4]$$

As $u$ ranges over the unit sphere, so does $v^T = u^TR$ and so instead of minimizing $F(u)$ over unit vectors $u$ we can instead work with the following equivalent optimization problem:

$$\min_{\|v\|_2=1} \mathbb{E}[(v^Tx)^4]$$

What are its local minima?

$$\mathbb{E}\left[(v^Tx)^4\right] = \mathbb{E}\left[\sum_i (v_ix_i)^4 + 6\sum_{ij}(v_ix_i)^2(v_jx_j)^2\right] =$$

$$= \sum_i v_i^4\,\mathbb{E}(x_i^4) + 6\sum_{ij} v_i^2v_j^2 + 3\sum_i v_i^4 - 3\sum_i v_i^4 + 3(\sum_i v_i^2)$$

$$= \sum_i v_i^4\left(\mathbb{E}\left[x_i^4\right] - 3\right) + 3$$

Hence the local minima of $F(v)$ correspond exactly to setting $v_i = \pm 1$ for some $i$. Recall that $v^T = u^T R$ and so this characterization implies that the local minima of $F(u)$ correspond to setting $u$ to be a column of $\pm R$.

The algorithm proceeds by using gradient descent (and a lower bound on the Hessian) to show that you can find a local optima of $F(u)$ quickly, and we can then recurse on the orthogonal complement to the vector we have found to find the other columns of $R$. This idea requires some care to show that the errors do not accumulate too badly, see [62], [116], [16].

In fact what we just computed are the *cumulants* that are an alternative basis for the moments of a distribution. Often these are much easier to work with since they satisfy the appealing property that the cumulants of the sum of independent variables $X_i$ and $X_j$ are the themselves the sum of the cumulants of $X_i$ and $X_j$. This is precisely the property we exploited here.

# Chapter 4

# Sparse Recovery

In this chapter we will study algorithms for sparse recovery: given a matrix $A$ and a vector $b$ that is a sparse linear combination of its columns – i.e. $Ax = b$ and $x$ is sparse – when can solve for $x$?

## 4.1 Basics

Throughout this section, we will consider only linear systems $Ax = b$ where $A$ has more columns than rows. Hence there is more than one solution for $x$ (if there is any solution at all), and we will be interested in finding the solution that has the smallest number of non-zeros:

**Definition 4.1.1** *Let $\|x\|_0$ be the number of non-zero entries of $x$.*

Unfortunately finding the sparsest solution to a system of linear equations in full generality is computationally hard, but there will be a number of important examples where we can solve for $x$ efficiently.

**Question 7** *When can we find for the sparsest solution to $Ax = b$?*

A trivial observation is that we can recover $x$ when $A$ has full column rank. In this case we can set $x = A^+ b$, where $A^+$ is the left-pseudo inverse of $A$. Note that this procedure works regardless of whether or not $x$ is sparse. In contrast, when $A$ has more columns than rows we will need to take advantage of the sparsity of $x$. We will show that under certain conditions on $A$, if $x$ is sparse enough then indeed it is the uniquely sparsest solution to $Ax = b$.

Our first goal is to prove that finding the sparsest solution to a linear system is hard. We will begin with the related problem:

**Problem 1 (P)** *Find the sparsest non-zero vector $x$ in a given subspace $S$*

Khachiyan [81] proved that this problem is $NP$-hard, and this result has many interesting applications that we will discuss later.

## Reduction from Subset Sum

We reduce from the following variant of subset sum:

**Problem 2 (S)** *Given distinct values $\alpha_1, \ldots, \alpha_m \in \mathbb{R}$, does there exist a set $I \subseteq [m]$ such that $|I| = n$ and $\sum_{i \in I} \alpha_i = 0$?*

We will embed an instance of this problem into the problem of finding the sparsest non-zero vector in a given subspace. We will make use of the following mapping which is called the *weird moment curve*:

$$\Gamma^w(\alpha_i) \Rightarrow \begin{bmatrix} 1 \\ \alpha_i \\ \alpha_i^2 \\ \ldots \\ \alpha_i^{n-2} \\ \alpha_i^n \end{bmatrix} \in \mathbb{R}^n$$

Note that this differs from the standard moment curve since the weird moment curve has $\alpha_i^n$ instead of $\alpha_i^{n-1}$.

**Claim 4.1.2** *A set $I$ with $|I| = n$ has $\sum_{i \in I} \alpha_i = 0$ if and only if the set of vectors $\{\Gamma^w(\alpha_i)\}_{i \in I}$ is linearly dependent.*

**Proof:** Consider the determinant of the matrix whose columns are $\{\Gamma^w(\alpha_i)\}_{i \in I}$. Then the proof is based on the following observations:

(a) The determinant is a polynomial in the variables $\alpha_i$ with total degree $\binom{n}{2} + 1$, which can be seen by writing the determinant in terms of its Laplace expansion (see e.g. [74]).

(b) Moreover the determinant is divisible by $\prod_{i<j} \alpha_i - \alpha_j$, since the determinant is zero if any $\alpha_i = \alpha_j$.

Hence we can write the determinant as

$$\Big(\prod_{\substack{i<j\\ i,j\in I}}(\alpha_i-\alpha_j)\Big)\Big(\sum_{i\in I}\alpha_i\Big)$$

We have assumed that the $\alpha_i$'s are distinct, and consequently the determinant is zero if and only if the sum of $\alpha_i = 0$. ■

We can now complete the proof that finding the sparsest non-zero vector in a subspace is hard: We can set $A$ to be the $n \times m$ matrix whose columns are $\Gamma^w(\alpha_i)$, and let $S = \ker(A)$. Then there is a vector $x \in S$ with $\|x\|_0 = n$ if and only if there is a subset $I$ with $|I| = n$ whose corresponding submatrix is singular. If there is no such set $I$ then any $x \in S$ has $\|x\|_0 > n$. Hence if we could find the sparsest non-zero vector in $S$ we could solve the above variant of subset sum.

In fact, this same proof immediately yields an interesting result in computational geometry (that was "open" several years after Khachiyan's paper).

**Definition 4.1.3** *A set of $m$ vectors in $\mathbb{R}^n$ is in* general position *if every set of at most $n$ vectors is linearly independent.*

From the above reduction we get that it is hard to decide whether a set of $m$ vectors in $\mathbb{R}^n$ is in general position or not (since there is an $I$ with $|I| = n$ whose submatrix is singular if and only if the vectors $\Gamma^w(\alpha_i)$ are not in general position).

Now we return to our original problem:

**Problem 3 (Q)** *Find the sparsest solution $x$ to $Ax = b$*

There is a subtle difference between **(P)** and **(Q)** since in **(P)** we restrict to *non-zero* vectors $x$ but in **(Q)** there is no such restriction on $x$. However there is a simple many-to-one reduction from **(Q)** to **(P)**.

**Lemma 4.1.4** *Finding the sparsest solution $x$ to $Ax = b$ is $NP$-hard.*

**Proof:** Suppose we are given a linear system $Ax = 0$ and we would like to find the sparsest non-zero solution $x$. Let $A^{-i}$ be equal to the matrix $A$ with he $i$th column deleted. Then for each $i$, let $x^{-i}$ be the sparsest solution to $A^{-i}x^{-i} = A_i$. Let $i^*$ be the index where $x^{-i}$ is the sparsest, and suppose $\|x^{-i}\|_0 = k$. We can build a solution $x$ to $Ax = 0$ with $\|x\|_0 = k + 1$ by setting the $i^*$th coordinate of $x$ to be $-1$. Indeed, it is not hard to see that $x$ is the sparsest solution to $Ax = 0$. ■

## 4.2 Uniqueness and Uncertainty Principles

### Incoherence

Here we will define the notion of an *incoherent* matrix $A$, and prove that if $x$ is sparse enough then it is the uniquely sparsest solution to $Ax = b$.

**Definition 4.2.1** *The columns of $A \in \mathbb{R}^{n\times m}$ are* $\mu$-incoherent *if for all $i \neq j$:*

$$|\langle A_i, A_j\rangle| \leq \mu \|A_i\| \cdot \|A_j\|$$

While the results we derive here can be extended to general $A$, we will restrict our attention to the case where $\|A_i\| = 1$, and hence a matrix is $\mu$-incoherent if for all $i \neq j$, $|\langle A_i, A_j\rangle| \leq \mu$.

In fact, incoherent matrices are quite common. Suppose we choose $m$ unit vetors at random in $\mathbb{R}^n$; then it is not hard to show that these vectors will be incoherent with $\mu = O(\sqrt{\frac{\log m}{n}})$. Hence even if $m = n^{100}$, these vectors will be $\widetilde{O}(1/\sqrt{n})$ incoherent. In fact, there are even better constructions of incoherent vectors that remove the logarithmic factors; this is almost optimal since for any $m > n$, any set of $m$ vectors in $\mathbb{R}^n$ has incoherence at least $\frac{1}{\sqrt{n}}$.

We will return to the following example several times: Consider the matrix $A = [I, D]$, where $I \in \mathbb{R}^{n\times n}$ is the identity matrix and $D \in \mathbb{R}^{n\times n}$ is the DFT matrix. In particular, $D_{ij} = \frac{w^{(i-1)(j-1)}}{\sqrt{n}}$ where $w = e^{i\frac{2\pi}{n}}$. This is often referred to as the spikes-and-sines matrix. It is not hard to see that $\mu = \frac{1}{\sqrt{n}}$ here.

### Uncertainty Principles

The important point is that if $A$ is incoherent, then if $x$ is sparse enough it will be the uniquely sparsest solution to $Ax = b$. These types of results were first established by the pioneering work of Donoho and Stark [53], and are based on establishing an *uncertainty principle.*

**Lemma 4.2.2** *Suppose we have $A = [U, V]$, where $U$ and $V$ are orthogonal. If $b = U\alpha = V\beta$, then $\|\alpha\|_0 + \|\beta\|_0 \geq \frac{2}{\mu}$.*

The interpretation of this result for the spikes-and-sines matrix is that any signal must have at least $\sqrt{n}$ non-zeros in the standard basis, or in the Fourier basis.

Informally, a signal cannot be too localized in both the time and frequency domains simultaneously!

**Proof:** Since $U$ and $V$ are orthonormal we have that $\|b\|_2 = \|\alpha\|_2 = \|\beta\|_2$. We can rewrite $b$ as either $U\alpha$ or $V\beta$ and hence $\|b\|_2^2 = |\beta^T(V^TU)\alpha|$. Because $A$ is incoherent, we can conclude that each entry of $V^TU$ has absolute value at most $\mu(A)$ and so $|\beta^T(V^TU)\alpha| \leq \mu(A)\|\alpha\|_1\|\beta\|_1$. Using Cauchy-Schwarz it follows that $\|\alpha\|_1 \leq \sqrt{\|\alpha\|_0}\|\alpha\|_2$ and thus

$$\|b\|_2^2 \leq \mu(A)\sqrt{\|\alpha\|_0\|\beta\|_0}\|\alpha\|_2\|\beta\|_2$$

Rearranging, we have $\frac{1}{\mu(A)} \leq \sqrt{\|\alpha\|_0\|\beta\|_0}$. Finally, applying the AM-GM inequality we get $\frac{2}{\mu} \leq \|\alpha\|_0 + \|\beta\|_0$ and this completes the proof. ■

Is this result tight? Indeed, returning to the spikes-and-sines example if choose $b$ to be the comb function, where the signal has equally spaced spikes at distance $\sqrt{n}$, then $b$ has $\sqrt{n}$ non-zeros in the standard basis. Moreover the comb function is its own discrete Fourier transform so it also has $\sqrt{n}$ non-zeros when represented using the Fourier basis.

Next, we apply the above uncertainty principle to prove a uniqueness result:

**Claim 4.2.3** *Suppose $A = [U, V]$ where $U$ and $V$ are orthonormal and $A$ is $\mu$-incoherent. If $Ax = b$ and $\|x\|_0 < \frac{1}{\mu}$, then $x$ is the uniquely sparsest solution.*

**Proof:** Consider any alternative solution $A\widetilde{x} = b$. Set $y = x - \widetilde{x}$ in which case $y \in \ker(A)$. Write $y$ as $y = [\alpha_y, \beta_y]^T$ and since $Ay = 0$, we have that $U\alpha_y = -V\beta_y$. We can now apply the uncertainty principle and conclude that $\|y\|_0 = \|\alpha_y\|_0 + \|\beta_y\|_0 \geq \frac{2}{\mu}$. It is easy to see that $\|\widetilde{x}\|_0 \geq \|y\|_0 - \|x\|_0 > \frac{1}{\mu}$ and so $\widetilde{x}$ has strictly more non-zeros than $x$ does, and this completes the proof. ■

Indeed, a similar statement is true even if $A$ is an arbitrary incoherent matrix (instead of a union of two orthonormal bases). We will discuss this extension further in the next section.

## Kruskal Rank

We can also work with a more general condition that is more powerful when proving uniqueness; however this condition is computationally hard to verify, unlike incoherence.

**Definition 4.2.4** *The* Kruskal rank *of a set of vectors $\{A_i\}_i$ is the maximum $r$ such that all subsets of $r$ vectors are linearly independent.*

In fact, we have already proven that it is $NP$-hard to compute the Kruskal rank of a given set of points, since deciding whether or not the Kruskal rank is $n$ is precisely the problem of deciding whether the points are in general position. Nevertheless, the Kruskal rank of $A$ is the right parameter for analyzing how sparse $x$ must be in order for it to be the uniquely sparest solution to $Ax = b$. Suppose the Kruskal rank of $A$ is $r$.

**Claim 4.2.5** *If $\|x\|_0 \leq r/2$ then $x$ is the unique sparsest solution to $Ax = b$.*

**Proof:** Consider any alternative solution $A\widetilde{x} = b$. Again, we can write $y = x - \widetilde{x}$ in which case $y \in \ker(A)$. However $\|y\|_0 \geq r + 1$ because every set of $r$ columns of $A$ is linearly independent, by assumption. Then $\|\widetilde{x}\|_0 \geq \|y\|_0 - \|x\|_0 \geq r/2 + 1$ and so $\widetilde{x}$ has strictly more non-zeros than $x$ does, and this completes the proof. ■

In fact, if $A$ is incoherent we can lower bound its Kruskal rank (and so the proof in the previous section can be thought of as a special case of the one in this).

**Claim 4.2.6** *If $A$ is $\mu$-incoherent then the Kruskal rank of the columns of $A$ is at least $1/\mu$.*

**Proof:** First we note that if there is a set $I$ of $r$ columns of $A$ that are linearly dependent, then the $I \times I$ submatrix of $A^TA$ must be singular. Hence it suffices to prove that every set $I$ of size $r$, the $I \times I$ submatrix of $A^TA$ is full rank for $r = 1/\mu$.

So consider any such a submatrix. The diagonals are one, and the off-diagonals have absolute value at most $\mu$ by assumption. We can now apply Gershgorin's disk theorem and conclude that the eigenvalues of the submatrix are strictly greater than zero provided that $r \leq 1/\mu$ (which implies that the sum of the absolute values of the off-diagonals in any row is strictly less than one). This completes the proof. ■

Hence we can extend the uniqueness result in the previous section to arbitrary incoherent matrices (instead of just ones that are the union of two orthonormal bases). Note that this bound differs from our earlier bound by a factor of two.

**Corollary 4.2.7** *Suppose $A$ is $\mu$-incoherent. If $Ax = b$ and $\|x\|_0 < \frac{1}{2\mu}$, then $x$ is the uniquely sparsest solution.*

There are a number of algorithms that recover $x$ up to the uniqueness threshold in the above corollary, and we will cover one such algorithm next.

## 4.3 Pursuit Algorithms

Here we will cover algorithms for solving sparse recovery when $A$ is incoherent. The first such algorithm is *matching pursuit* and was introduced by Mallat and Zhang [93]; we will instead analyze *orthogonal matching pursuit* [99]:

```
Orthogonal Matching Pursuit
Input: matrix $A \in \mathbb{R}^{n\times m}$, vector $b \in \mathbb{R}^n$, desired number of nonzero entries $k \in \mathbb{N}$.
Output: solution $x$ with at most $k$ nonzero entries.

Initialize: $x^0 = 0$, $r^0 = Ax^0 - b$, $S = \emptyset$.
For $\ell = 1, 2, \dots, k$
    Choose column $j$ that maximizes $\frac{|\langle A_j, r^{\ell-1}\rangle|}{\|A_j\|_2^2}$.
Add $j$ to $S$.
    Set $r^\ell = \mathrm{proj}_{U^\perp}(b)$, where $U = \mathrm{span}(A_S)$.
    If $r^\ell = 0$, break.
End
Solve for $x_S$: $A_S x_S = b$. Set $x_{\bar{S}} = 0$.
```

Let $A$ be $\mu$-incoherent and suppose that there is a solution $x$ with $k < 1/(2\mu)$ nonzero entries, and hence $x$ is the uniquely sparsest solution to the linear system. Let $T = \mathrm{supp}(x)$. We will prove that orthogonal matching pursuit recovers the true solution $x$. Our analysis is based on establishing the following two invariants for our algorithm:

(a) Each index $j$ the algorithm selects is in $T$ .

(b) Each index $j$ gets chosen at most once.

These two properties immediately imply that orthogonal matching pursuit recovers the true solution $x$, because the residual error $r^\ell$ will be non-zero until $S = T$, and moreover the linear system $A_T x_T = b$ has a unique solution (since otherwise $x$ would not be the uniquely sparsest solution, which contradicts the uniqueness property that we proved in the previous section).

Property **(b)** is straightforward, because once $j \in S$ at every subsequent step in the algorithm we will have that $r^\ell \perp U$, where $U = \mathrm{span}(A_S)$, so $\langle r^\ell, A_j\rangle = 0$ if $j \in S$. Our main goal is to establish property **(a)**, which we will prove inductively. The main lemma is:

**Lemma 4.3.1** *If $S \subseteq T$ at the start of a stage, then the algorithm selects $j \in T$.*

We will first prove a helper lemma:

**Lemma 4.3.2** *If $r^{\ell-1}$ is supported in $T$ at the start of a stage, then the algorithm selects $j \in T$.*

**Proof:** Let $r^{\ell-1} = \sum_{i \in T} y_i A_i$. Then we can reorder the columns of $A$ so that the first $k'$ columns correspond to the $k'$ nonzero entries of $y$, in decreasing order of magnitude:

$$\underbrace{|y_1| \geq |y_2| \geq \cdots \geq |y_{k'}| > 0}_{\text{corresponds to first } k' \text{ columns of } A}, \quad |y_{k'+1}| = 0, |y_{k'+2}| = 0, \ldots, |y_m| = 0.$$

where $k' \leq k$. Hence $\text{supp}(y) = \{1, 2, \ldots, k'\} \subseteq T$. Then to ensure that we pick $j \in T$, a sufficient condition is that

$$|\langle A_1, r^{\ell-1} \rangle| > |\langle A_i, r^{\ell-1} \rangle| \qquad \text{for all } i \geq k'+1. \tag{4.1}$$

We can lower-bound the left-hand side of (4.1):

$$|\langle r^{\ell-1}, A_1 \rangle| = \left| \Big\langle \sum_{\ell=1}^{k'} y_\ell A_\ell, A_1 \Big\rangle \right| \geq |y_1| - \sum_{\ell=2}^{k'} |y_\ell| |\langle A_\ell, A_1 \rangle| \geq |y_1| - |y_1|(k'-1)\mu \geq |y_1|(1 - k'\mu + \mu),$$

which, under the assumption that $k' \leq k < 1/(2\mu)$, is strictly lower-bounded by $|y_1|(1/2 + \mu)$.

We can then upper-bound the right-hand side of (4.1):

$$|\langle r^{\ell-1}, A_i \rangle| = \left| \Big\langle \sum_{\ell=1}^{k'} y_\ell A_\ell, A_i \Big\rangle \right| \leq |y_1| \sum_{\ell=1}^{k'} |\langle A_\ell, A_i \rangle| \leq |y_1| k'\mu,$$

which, under the assumption that $k' \leq k < 1/(2\mu)$, is strictly upper-bounded by $|y_1|/2$. Since $|y_1|(1/2 + \mu) > |y_1|/2$, we conclude that condition (4.1) holds, guaranteeing that the algorithm selects $j \in T$ and this completes the proof of the lemma. ■

Now we can prove the main lemma:

**Proof:** Suppose that $S \subseteq T$ at the start of a stage. Then the residual $r^{\ell-1}$ is supported in $T$ because we can write it as

$$r^{\ell-1} = b - \sum_{i \in S} z_i A_i, \text{ where } z = \arg\min \|b - A_S z_S\|_2$$

Applying the above lemma, we conclude that the algorithm selects $j \in T$. ■

This establishes property **(a)** inductively, and completes the proof of correctness for orthogonal matching pursuit. Note that this algorithm works up to exactly the threshold where we established uniqueness. However in the case where $A = [U, V]$ and $U$ and $V$ are orthogonal, we proved a uniqueness result that is better by a factor of two. There is no known algorithm that matches the best known uniqueness bound there, although there are better algorithms than the one above (see e.g. [55]).

### Matching Pursuit

We note that matching pursuit differs from orthogonal matching pursuit in a crucial way: In the latter, we recompute the coefficients $x_i$ for $i \in S$ at the end of each stage because we project $b$ perpendicular to $U$. However we could hope that these coefficients do not need to be adjusted much when we add a new index $j$ to $S$. Indeed, matching pursuit does not recompute these coefficients and hence is faster in practice, however the analysis is more involved because we need to keep track of how the error accumulates.

## 4.4 Prony's Method

The main result in this section is that any $k$-sparse signal can be recovered from just the first $2k$ values of its discrete Fourier transform, which has the added benefit that we can compute $Ax$ quickly using the FFT. This is algorithm is called Prony's method, and dates back to 1795. This is optimal optimal relationship between the number of rows in $A$ and the bound on the sparsity of $x$; however this method is very unstable since it involves inverting a Vandermonde matrix.

### Properties of the DFT

In Prony's method, we will make crucial use of some of the properties of the DFT. Recall that DFT matrix has entries:

$$F_{a,b} = \left(\frac{1}{\sqrt{n}}\right) \exp\left(\frac{i2\pi(a-1)(b-1)}{n}\right)$$

We can write $\omega = e^{i2\pi/n}$, and then the first row is $\frac{1}{\sqrt{n}}[1, 1, \ldots, 1]$; the second row is $\frac{1}{\sqrt{n}}[1, \omega, \omega^2, \ldots]$, etc.

We will make use of following basic properties of $F$:

(a) $F$ is orthonormal: $F^H F = FF^H$, where $F^H$ is the Hermitian transpose of $F$

(b) $F$ diagonalizes the convolution operator

In particular, we will define the convolution operation through its corresponding linear transformation:

**Definition 4.4.1 (Circulant matrix)** *For a vector $c = [c_1, c_2, \ldots, c_n]$, let*

$$M^c = \begin{bmatrix} c_n & c_{n-1} & c_{n-2} & \ldots & c_1 \\ c_1 & c_n & c_{n-1} & \ldots & c_2 \\ \vdots & & & & \vdots \\ c_{n-1} & \ldots & \ldots & \ldots & c_n \end{bmatrix}.$$

Then we can define $M^c x$ as the result of convolving $c$ and $x$, denoted by $c * x$. It is easy to check that this coincides with the standard definition of convolution.

In fact, we can diagonalize $M^c$ using $F$. We will use the following fact, without proof:

**Claim 4.4.2** *$M^c = F^H diag(\widehat{c})F$, where $\widehat{c} = Fc$.*

Hence we can think about convolution as coordinate-wise multiplication in the Fourier representation:

**Corollary 4.4.3** *Let $z = c * x$; then $\widehat{z} = \widehat{c} \odot \widehat{x}$, where $\odot$ indicates coordinate-wise multiplication.*

**Proof:** We can write $z = M^c x = F^H \text{diag}(\widehat{c}) F x = F^H \text{diag}(\widehat{c})\widehat{x} = F^H(\widehat{c} \odot \widehat{x})$, and this completes the proof. ■

We introduce the following helper polynomial, in order to describe Prony's method:

**Definition 4.4.4 (Helper polynomial)**

$$\begin{aligned} p(z) &= \prod_{b \in \text{supp}(x)} \omega^{-b}(\omega^b - z) \\ &= 1 + \lambda_1 z + \ldots + \lambda_k z^k. \end{aligned}$$

**Claim 4.4.5** *If we know $p(z)$, we can find $supp(x)$.*

**Proof:** In fact, an index $b$ is in the support of $x$ if and only if $p(\omega^b) = 0$. So we can evaluate $p$ at powers of $\omega$, and the exponents where $p$ evaluates to a non-zero are exactly the support of $x$. ■

The basic idea of Prony's method is to use the first $2k$ values of the discrete Fourier transform to find $p$, and hence the support of $x$. We can then solve a linear system to actually find the values of $x$.

## Finding the Helper Polynomial

Our first goal is to find the Helper polynomial. Let

$$v = [1, \lambda_1, \lambda_2, \ldots, \lambda_k, 0, \ldots, 0], \text{ and } \widehat{v} = Fv$$

It is easy to see that the value of $\widehat{v}$ at index $b+1$ is exactly $p(\omega^b)$.

**Claim 4.4.6** *$supp(\widehat{v}) = \overline{supp(x)}$*

That is, the zeros of $\widehat{v}$ correspond roots of $p$, and hence non-zeros of $x$. Conversely, the non-zeros of $\widehat{v}$ correspond to zeros of $x$. We conclude that $x \odot \widehat{v} = 0$, and so:

**Corollary 4.4.7** $M^{\widehat{x}} v = 0$

**Proof:** We can apply Claim 4.4.2 to rewrite $x \odot \widehat{v} = 0$ as $\widehat{x} * v = \widehat{0} = 0$, and this implies the corollary. ■

Let us write out this linear system explicitly:

$$M^{\widehat{x}} = \begin{bmatrix} \widehat{x}_n & \widehat{x}_{n-1} & \ldots & \widehat{x}_{n-k} & \ldots & \widehat{x}_1 \\ \widehat{x}_1 & \widehat{x}_n & \ldots & \widehat{x}_{n-k+1} & \ldots & \widehat{x}_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \widehat{x}_{k+1} & \widehat{x}_k & \ldots & \widehat{x}_1 & \ldots & \widehat{x}_{k+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \widehat{x}_{2k} & \widehat{x}_{2k-1} & \ldots & \widehat{x}_k & \ldots & \widehat{x}_{2k+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Recall, we do not have access to all of the entries of this matrix since we are only given the first $2k$ values of the DFT of $x$. However consider the $k \times k+1$ submatrix whose top left value is $\widehat{x}_{k+1}$ and whose bottom right value is $\widehat{x}_k$. This matrix only involves the values that we do know!

Consider

$$\begin{bmatrix} \widehat{x}_k & \widehat{x}_{k-1} & \dots & \widehat{x}_1 \\ \vdots & & & \\ \widehat{x}_{2k-1} & \widehat{x}_{2k-1} & \dots & \widehat{x}_k \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_k \end{bmatrix} = - \begin{bmatrix} \widehat{x}_{k+1} \\ \vdots \\ \vdots \\ \widehat{x}_{2k} \end{bmatrix}$$

It turns out that this linear system is full rank, so $\lambda$ is the unique solution to the linear system (the proof is left to the reader[1]). The entries in $\lambda$ are the coefficients of $p$, so once we have solved for $\lambda$ we can evaluate the helper polynomial on $\omega^b$ to find the support of $x$. All that remains is to find the values of $x$. Indeed, let $M$ be the restriction of $F$ to the columns in $S$ and its first $2k$ rows. $M$ is a Vandermonde matrix, so again $Mx_S = \widehat{x}_{1,2,\dots 2k}$ has a unique solution, and we can solve this linear system to find the non-zero values of $x$.

## 4.5 Compressed Sensing

Here we will give *stable* algorithms for recovering a signal $x$ that has an almost linear (in the number of rows of the sensing matrix) number of non-zeros. Recall that the Kruskal rank of the columns of $A$ is what determines how many non-zeros we can allow in $x$ and yet have $x$ be the uniquely sparsest solution to $Ax = b$. A random matrix has large Kruskal rank, and what we will need for compressed sensing is a robust analogue of Kruskal rank:

**Definition 4.5.1** *A matrix $A$ is RIP with constant $\delta_k$ if for all $k$-sparse vectors $x$ we have:*

$$(1 - \delta_k)\|x\|_2^2 \le \|Ax\|_2^2 \le (1 + \delta_k)\|x\|_2^2$$

If $A$ is a random $m \times n$ matrix where each entry is an independent Gaussian ($N(0,1)$) then we can choose $m \approx k \log n/k$ and $\delta_k = 1/3$. Next we formalize the goals of sparse recovery:

**Definition 4.5.2** $\sigma_k(x) = \min_{w \ s.t. \ \|w\|_0 \le k} \|x - w\|_1$

i.e. $\sigma_k(x)$ is the $\ell_1$ sum of all but the $k$ largest entries of $x$. In particular, if $\|x\|_0 \le k$ then $\sigma_k(x) = 0$.

Our goal is to find a $w$ where $\|x - w\|_1 \leq C\sigma_k(x)$ from a few ($\tilde{O}(k)$) measurements. Note that we will note require that $w$ is $k$ sparse. However if $x$ is exactly $k$ sparse, then any $w$ satisfying the above condition must be exactly equal to $x$ and hence this new recovery goal subsumes our exact recovery goals from previous lectures (and is indeed much stronger).

The natural (but intractable) approach is:

$$(P0) \qquad \min \|w\|_0 \text{ s.t. } Aw = b$$

Since this is computationally hard to solve (for all $A$) we will work with the $\ell_1$ relaxation:

$$(P1) \qquad \min \|w\|_1 \text{ s.t. } Aw = b$$

and we will prove conditions under which the solution $w$ to this optimization problem satisfies $w = x$ (or $\|x - w\|_1 \leq C\sigma_k(x)$).

**Theorem 4.5.3** *[35] If $\delta_{2k} + \delta_{3k} < 1$ then if $\|x\|_0 \leq k$ we have $w = x$.*

**Theorem 4.5.4** *[34] If $\delta_{3k} + 3\delta_{4k} < 2$ then*

$$\|x - w\|_2 \leq \frac{C}{\sqrt{k}}\sigma_k(x)$$

Note that this bounds the $\ell_2$ norm of the error $x - w$ in terms of the $\ell_1$ error of the best approximation.

**Theorem 4.5.5** *[40] If $\delta_{2k} < 1/3$ then*

$$\|x - w\|_1 \leq \frac{2 + 2\delta_{2k}}{1 - 3\delta_{2k}}\sigma_k(x)$$

We will follow the proof of [80] that greatly stream-lined the types of analysis and made the connection between compressed sensing and *almost Euclidean subsections* explicit. From this viewpoint it will be much easier to draw an analogy with error correcting codes.

## Almost Euclidean Subsections

Set $\Gamma = \ker(A)$. We will make use of certain geometric properties of $\Gamma$ (that hold almost surely) in order to prove that basis pursuit works:

**Definition 4.5.6** *$\Gamma \subseteq \mathbb{R}^n$ is an almost Euclidean subsection if for all $v \in \Gamma$,*

$$\frac{1}{\sqrt{n}}\|v\|_1 \leq \|v\|_2 \leq \frac{C}{\sqrt{n}}\|v\|_1$$

Note that the inequality $\frac{1}{\sqrt{n}}\|v\|_1 \leq \|v\|_2$ holds for all vectors, hence the second inequality is the important part. What we are requiring is that the $\ell_1$ and $\ell_2$ norms are almost equivalent after rescaling.

**Question 8** *If a vector $v$ has $\|v\|_0 = o(n)$ then can $v$ be in $\Gamma$?*

No! Any such vector $v$ would have $\|v\|_1 = o(\sqrt{n})\|v\|_2$ using Cauchy-Schwartz.

Let us think about these subsections geometrically. Consider the unit ball for the $\ell_1$ norm:

$$B_1 = \{v | \|v\|_1 \leq 1\}$$

This is called the cross polytope and is the convex hull of the vectors $\{\pm e_i\}_i$ where $e_i$ are the standard basis vectors. Then $\Gamma$ is a subspace which when intersected with $B_1$ results in a convex body that is close to the sphere $B_2$ after rescaling.

In fact it has been known since the work of [63] that choosing $\Gamma$ uniformly at random with $\dim(\Gamma) \geq n - m$ we can choose $C = O(\sqrt{\log n/m})$ almost surely (in which case it is the kernel of an $m \times n$ matrix $A$, which will be our sensing matrix). In the remainder of the lecture, we will establish various geometric properties of $\Gamma$ that will set the stage for compressed sensing.

## Properties of $\Gamma$

Throughout this section, let $S = n/C^2$.

**Claim 4.5.7** *Let $v \in \Gamma$, then either $v = 0$ or $|supp(v)| \geq S$.*

**Proof:**

$$\|v\|_1 = \sum_{j \in \text{supp}(v)} |v_j| \leq \sqrt{|\text{supp}(v)|} \cdot \|v\|_2 \leq \sqrt{|\text{supp}(v)|}\frac{C}{\sqrt{n}}\|v\|_1$$

where the last inequality uses the property that $\Gamma$ is almost Euclidean. The last inequality implies the claim. ■

Now we can draw an analogy with error correcting codes. Recall that here we want $\mathcal{C} \subseteq \{0,1\}^n$. And the rate $R$ is $R = \log|\mathcal{C}|/n$ and the relative distance $\delta$ is

$$\delta = \frac{\min_{x \neq y \in \mathcal{C}} d_H(x,y)}{n}$$

where $d_H$ is the Hamming distance. The goal is to find a code where $R, \delta = \Omega(1)$ and that are easy to encode and decode. In the special case of linear codes, e.g. $\mathcal{C} = \{y | y = Ax\}$ where $A$ is an $n \times Rn$ $\{0,1\}$-valued matrix and $x \in \{0,1\}^{Rn}$. Then

$$\delta = \frac{\min_{x \neq 0 \in \mathcal{C}} \|x\|_0}{n}$$

So for error correcting codes we want to find large (linear) dimensional subspaces where each vector has a linear number of non-zeros. In compressed sensing we want $\Gamma$ to have this property too, but moreover we want that its $\ell_1$ norm is also equally spread out (e.g. most of the non-zero coordinates are large).

**Definition 4.5.8** *For $\Lambda \subseteq [n]$, let $v_\Lambda$ denote the restriction of $v$ to coordinates in $\Lambda$. Similarly let $v^\Lambda$ denote the restriction of $v$ to $\bar{\Lambda}$.*

**Claim 4.5.9** *Suppose $v \in \Gamma$ and $\Lambda \subseteq [n]$ and $|\Lambda| < S/16$. Then*

$$\|v_\Lambda\|_1 < \frac{\|v\|_1}{4}$$

**Proof:**

$$\|v_\Lambda\|_1 \leq \sqrt{|\Lambda|}\|v_\Lambda\|_2 \leq \sqrt{|\Lambda|}\frac{C}{\sqrt{n}}\|v\|_1$$

■

Hence not only do vectors in $\Gamma$ have a linear number of non-zeros, but in fact their $\ell_1$ norm is spread out. Now we are ready to prove that $(P1)$ exactly recovers $x$ when $\|x\|_0$ is sufficiently small (but nearly linear). Next lecture we will prove that it is also stable (using the properties we have established for $\Gamma$ above).

**Lemma 4.5.10** *Let $w = x + v$ and $v \in \Gamma$ where $\|x\|_0 \leq S/16$. Then $\|w\|_1 > \|x\|_1$.*

**Proof:** Set $\Lambda = \text{supp}(x)$.

$$\|w\|_1 = \|(x+v)_\Lambda\|_1 + \|(x+v)^\Lambda\|_1 = \|(x+v)_\Lambda\|_1 + \|v^\Lambda\|_1$$

Now we can invoke triangle inequality:

$$\|w\|_1 \geq \|x_\Lambda\|_1 - \|v_\Lambda\|_1 + \|v^\Lambda\|_1 = \|x\|_1 - \|v_\Lambda\|_1 + \|v^\Lambda\|_1 = x_\Lambda\|_1 - 2\|v_\Lambda\|_1 + \|v\|_1$$

However $\|v\|_1 - 2\|v_\Lambda\|_1 \geq \|v\|_1/2 > 0$ using the above claim. This implies the lemma. ■

Hence we can use almost Euclidean subsections to get exact sparse recovery up to

$$\|x\|_0 = S/16 = \Omega(n/C^2) = \Omega\Big(\frac{n}{\log n/m}\Big)$$

Next we will consider stable recovery. Our main theorem is:

**Theorem 4.5.11** *Let $\Gamma = ker(A)$ be an almost Euclidean subspace with parameter $C$. Let $S = \frac{n}{C^2}$. If $Ax = Aw = b$ and $\|w\|_1 \leq \|x\|_1$ we have*

$$\|x - w\|_1 \leq 4\,\sigma_{\frac{S}{16}}(x)\,.$$

**Proof:** Let $\Lambda \subseteq [n]$ be the set of $S/16$ coordinates maximizing $\|x_\Lambda\|_1$. We want to upper bound $\|x - w\|_1$. By the repeated application of the triangle inequality, $\|w\|_1 = \|w^\Lambda\|_1 + \|w_\Lambda\|_1 \leq \|x\|_1$ and the definition of $\sigma_t(\cdot)$, it follows that

$$\begin{aligned}
\|x - w\|_1 &= \|(x-w)_\Lambda\|_1 + \|(x-w)^\Lambda\|_1 \\
&\leq \|(x-w)_\Lambda\|_1 + \|x^\Lambda\|_1 + \|w^\Lambda\|_1 \\
&\leq \|(x-w)_\Lambda\|_1 + \|x^\Lambda\|_1 + \|x\|_1 - \|w_\Lambda\|_1 \\
&\leq 2\|(x-w)_\Lambda\|_1 + 2\|x^\Lambda\|_1 \\
&\leq 2\|(x-w)_\Lambda\|_1 + 2\,\sigma_{\frac{S}{16}}(x)\,.
\end{aligned}$$

Since $(x - w) \in \Gamma$, we can apply Claim 4.5.9 to conclude that $\|(x-w)_\Lambda\|_1 \leq \frac{1}{4}\|x - w\|_1$. Hence

$$\|x - w\|_1 \leq \frac{1}{2}\|x - w\|_1 + 2\sigma_{\frac{S}{16}}(x)\,.$$

This completes the proof. ■

Notice that in the above argument, it was the geometric properties of $\Gamma$ which played the main role. There are a number of proofs that basis pursuit works, but the advantage of the one we presented here is that it clarifies the connection between the classical theory of error correction over the finite fields, and the sparse recovery. The matrix $A$ here plays the role *parity check matrix* of error correcting code, and hence its kernel corresponds to the *codewords*. There is more subtlety in the real case though: as opposed to the finite field setting where the Hamming distance is essentially the only reasonable way of measuring the magnitude of errors, in the real case there is an interplay among many different norms, giving rise to some phenomenon not present in the finite field case.

In fact, one of the central open question of the field is to give a *deterministic* construction of RIP matrices:

**Question 9 (Open)** *Is there an explicit construction of RIP matrices, or equivalently an almost Euclidean subsection $\Gamma$?*

In contrast, there are many explicit constructions of asymptotically good codes. The best known deterministic construction is due to Guruswami, Lee and Razborov:

**Theorem 4.5.12** *[69] There is a polynomial time deterministic algorithm for constructing an almost Euclidean subspace $\Gamma$ with parameter $C \sim (\log n)^{\log\log\log n}$*

We note that it is easy to achieve weaker guarantees, such as $\forall 0 \neq v \in \Gamma$, $\mathrm{supp}(v) = \Omega(n)$, but these do not suffice for compressed sensing since we also require that the $\ell_1$ weight is spread out too.

# Chapter 5

# Dictionary Learning

In this chapter we will study *dictionary learning*, where we are given many examples $b_1, b_2, ..., b_p$ and we would like to find a dictionary $A$ so that every example can be written as a sparse linear combination of the columns in $A$.

## 5.1 Background

In the previous chapter, we studied algorithms for *sparse recovery*. However in many applications, the dictionary $A$ is unknown and has to be learned from examples. An alternative motivation is that often one hand designs families of features to serve as an appropriate dictionary, e.g. sinusoids, wavelets, ridgelets, curvelets, etc. [94] and each one is useful in different settings: wavelets for impulsive events, ridgelets for discontinuities in edges, curvelets for smooth curves, etc. But given a new collection of data, can we learn the right families to represent the signal automatically?

Recall that it is $NP$-hard to find the sparsest solution to an *arbitrary* linear system. However there are important classes of problems for which the problem can be solved efficiently. Similarly, we cannot hope to solve the dictionary learning problem for an arbitrary dictionary. We will instead focus on designing algorithms for dictionary learning in the settings where we already know that we can solve the corresponding sparse recovery problem too. After all, finding a dictionary is only useful if you can use it. The three most important cases where we can do sparse recovery are:

(a) $A$ has full column rank

(b) $A$ is incoherent

(c) $A$ is RIP

We will present an algorithm of Spielman, Wang and Wright [108] that succeeds (under reasonable distributional assumptions) if $A$ is full rank, and if each $b_i$ is a linear combination of at most $\widetilde{O}(\sqrt{n})$ columns in $A$. Next, we will give an algorithm of Arora, Ge and Moitra [15] for the incoherent and overcomplete case that also succeeds up to $\widetilde{O}(\sqrt{n})$ sparsity. We note that Agarwal et al [2], [3] recently and independently gave alternative algorithms for dictionary learning that work up to a weaker sparsity bound of $\widetilde{O}(n^{1/4})$.

## History

The dictionary learning problem was introduced by Olshausen and Field [97], who were interested in understanding various properties of the mammalian visual cortex. Neuroscientists often measure the *receptive field* of neurons – namely, how the neurons respond to various types of stimuli. Indeed, their response is well-known to be *localized*, *bandpass* and *oriented* and Olshausen and Field observed that if one learns a dictionary for a collection of natural images, the elements of the dictionary often have many of these same properties. Their work suggested that an important step in understanding the visual system could be in identifying the basis it uses to represent natural images.

Dictionary learning, or as it is often called *sparse coding*, is a basic building block of many machine learning systems. This algorithmic primitive arises in applications ranging from denoising, edge-detection, super-resolution and compression. Dictionary learning is also used in the design of some deep learning architectures. Popular approaches to solving this problem in practice are variants of the standard alternating minimization approach. Suppose the pairs $(x_i, b_i)$ are collected into the columns of matrices $X$ and $B$ respectively. Then our goal is to compute $A$ and $X$ from $B$ in such a way that the columns of $X$ are sparse.

**Method of Optimal Directions [56]** : Start with an initial guess for $A$, and then alternately update either $A$ or $X$:

- Given $A$, compute a sparse $X$ so that $AX \approx B$ (using e.g. matching pursuit [93] or basis pursuit [39])
- Given $X$, compute the $A$ that minimizes $\|AX - B\|_F$

This algorithm converges to a local optimum, because in each step the error $\|AX - Y\|_F$ will only decrease.

**K-SVD [5]** Start with an initial guess for $A$. Then repeat the following procedure:

- Given $A$, compute a sparse $X$ so that $AX \approx B$ (again, using a pursuit method)
- Group all data points $B^{(1)}$ where the corresponding $X$ vector has a non-zero at index $i$. Subtract off components in the other directions
$$B^{(1)} - \sum_{j \neq i} A_j X_j^{(1)}$$
- Compute the first singular vector $v_1$ of the residual matrix, and update the column $A_i$ to $v_1$

In practice, these algorithms are highly sensitive to their starting conditions. In the next sections, we will present alternative approaches to dictionary learning with provable guarantees. Recall that some dictionaries are easy to solve sparse recovery problems for, and some are hard. Then an important rationale for preferring provable algorithms here is that if the examples are generated from an easy dictionary, our algorithms will really learn an easy dictionary (but for the above heuristics this need not be the case).

## 5.2 Full Rank Dictionaries

Here we present a recent algorithm of Spielman, Wang and Wright [108] that works when $A$ has full column rank and $X$ has $\widetilde{O}(\sqrt{n})$ non-zeros, under certain distributional conditions. First we define the model:

### The Model

The distributional model in [108] is in fact more general than the one we will work with here; we do this merely to simplify the analysis and notation but the proof will be nearly identical. Let $\theta \in [\frac{1}{n}, \frac{1}{\sqrt{n \log n}}]$ be the sparsity parameter. Then the vectors $x_i$ are generated according to the following procedure:

(a) Each coordinate of $x_i$ is chosen (independently) to be nonzero with probability $\theta$

(b) If the $j$th coordinate is non-zero, then its value is sampled (independently) from a standard Gaussian $\mathcal{N}(0, 1)$.

We observe samples $b_i = Ax_i$. The dictionary $A$ is an unknown invertible $n \times n$ matrix, and we are asked to recover $A$ exactly from a polynomial number of samples of the form $b_i$.

**Theorem 5.2.1** *[108] There is a polynomial time algorithm to learn a full-rank dictionary $A$ exactly given a polynomial number of samples from the above model.*

Strictly speaking, if the coordinates of $x_i$ are perfectly independent then we could recover $A$ alternatively using provable algorithms for *independent component analysis* [62], but the dictionary learning algorithms work more generally even when the coordinates are not independent.

## Outline

The basic idea is to consider the row space of $B$ which we will denote by $R = \{w^T B\}$. Note that $A^{-1}B = X$ and hence the rows of $X$ are contained in $R$. The crucial observation is:

**Observation 5.2.2** *The rows of $X$ are the sparsest non-zero vectors in $R$.*

Of course finding the sparsest non-zero vector in a subspace is hard, so we will instead consider an $\ell_1$-relaxation similar to what we did for sparse recovery. Consider the following four optimization problems:

$$(P0) \qquad \min \|w^T B\|_0 \text{ s.t. } w \neq 0$$

This optimization problem asks for the sparsest (non-zero) vector in $R$. However it is $NP$-hard. We instead consider:

$$(P1) \qquad \min \|w^T B\|_1 \text{ s.t. } r^T w = 1$$

The constraint $r^T w$ fixes a normalization, and we will explain how to choose $r$ later. The above optimization problem can be solve efficiently because it is a linear program. We would like to show that its optimal solution $w^T B$ is a (scaled) row of $X$. In fact we can transform the above linear program into a simpler one that will be easier to analyze:

$$(Q1) \qquad \min \|z^T X\|_1 \text{ s.t. } c^T z = 1$$

There is a bijection between the solutions of this linear program and those of $(P1)$ given by $z = A^T w$ and $c = A^{-1} r$. Let us set $r$ to be a column of $B$ and hence $c$ is the corresponding column of $X$ and is sparse. Now we can explain the intuition:

- We will prove that the solution to $(Q1)$ is sparse, in particular $\mathrm{supp}(z) \subseteq \mathrm{supp}(c)$.

- And for sparse $z$ we will have that $\|z^T X\|_1 \approx \|z\|_1$ (after an appropriate scaling). Hence we can instead analyze the linear program:

$$(Q1') \qquad \min \|z\|_1 \text{ s.t. } c^T z = 1$$

Note that $|\mathrm{supp}(z)| = 1$ if $c$ has a coordinate that is strictly the largest in absolute value. Recall that we chose $c$ to be a column of $X$, and in our distributional model the non zeros in $X$ are Gaussian. Hence almost surely $c$ will have a strictly largest coordinate $c_i$ in absolute value, and if we solve $(P1)$ the vector in the objective function (namely $w^T B$) will be the $i$th row of $X$ up to rescaling.

- Lastly we can repeatedly solve $(P1)$ and find all of the rows of $X$ and after correcting their scaling we can solve for $A$ by computing $A = BX^+$.

## Step 1

Suppose that $z_*$ is an optimal solution to $(Q1)$, where $c = x_i$ is a column of $X$. Set $J = \mathrm{supp}(c)$, we can write $z_* = z_0 + z_1$ where $z_0$ is supported in $J$ and $z_1$ is supported in $\overline{J}$. Note that $c^T z_0 = c^T z_*$. We would like to show that $z_0$ is at least as good of a solution to $(Q1)$ as $z_*$ is. In particular we want to prove that

$$\|z_0^T X\|_1 < \|z_*^T X\|_1$$

**Definition 5.2.3** *If $R$ is a set of rows and $C$ is a set of columns, let $X_C^R$ be the submatrix of $X$ that is the intersection of those rows and columns.*

Let $S$ be the set of columns of $X$ that have a non-zero entry in $J$. That is $S = \{j | X_j^J \neq \vec{0}\}$. We now compute:

$$\begin{aligned}\|z_*^T X\|_1 &= \|z_*^T X_S\|_1 + \|z_*^T X_{\overline{S}}\|_1 \\ &\geq \|z_0^T X_S\|_1 - \|z_1^T X_S\|_1 + \|z_1^T X_{\overline{S}}\|_1 \\ &\geq \|z_0^T X\|_1 - 2\|z_1^T X_S\|_1 + \|z_1^T X\|_1\end{aligned}$$

It remains to show that

$$\|z_1^T X\|_1 - 2\|z_1^T X_S\|_1 > 0 \tag{5.1}$$

Let us suppose that $z_1$ is fixed (we can apply a union bound over an $\varepsilon$-net of this space to complete the argument). Then $S$ is a random set and we can compute:

$$\mathbb{E}[\|z_1^T X_S\|_1] = \frac{|S|}{p}\,\mathbb{E}[\|z_1^T X\|_1]$$

The expected size of $S$ is $p \times \mathbb{E}[|\text{supp}(x_i)|] \times \theta = \theta^2 np = o(p)$. Together, these imply that

$$\mathbb{E}[\|z_1^T X\|_1 - 2\|z_1^T X_S\|_1] = \left(1 - \frac{2\,\mathbb{E}[|S|]}{p}\right)\mathbb{E}[\|z_1^T X\|_1]$$

is bounded away from zero, and thus (5.1) holds with high probability for any fixed $z_1$. We can take a union bound over an $\varepsilon$-net of all possible $z_1$'s and conclude that (5.1) holds for all $z_1$'s. This in turn implies that if $z_1$ is non-zero then $\|z_0^T X\|_1 < \|z_*^T X\|_1$ but this is a contradiction since we assumed that $z_*$ is an optimal solution to $(Q1)$. We conclude that $z_1$ is zero and so $\text{supp}(z_*) \subseteq \text{supp}(c)$, as desired.

## Step 2

We wish to show that:

$$\|z^T X\|_1 \approx \|z^T\|_1$$

up to some scaling factor provided that $\text{supp}(z)$ is small enough. Recall from the previous step we know that $\text{supp}(z) \subseteq \text{supp}(c) = J$ and $|J| \leq \widetilde{O}(\theta n)$. Note that the typical column of $X^J$ has $\theta|J| = \theta^2 n = o(1)$ nonzero entries in expectation. That means that, with high probability, the vast majority of the columns in $X^J$ have at most one non-zero entry. It is easy to see that:

**Claim 5.2.4** *If each column of $X^J$ has at most one non-zero entry, then*

$$\mathbb{E}[\|z_J^T X^J\|_1] = C\frac{p}{|J|}\|z_J\|_1$$

*where $C$ is the expected absolute value of a non-zero in $X$.*

So we can establish Step 2 by bounding the contribution of columns of $X^J$ with two or more non-zeros. Hence this completes the proof. (The more concise description of this step is that since $|J|$ is $\sqrt{n}$ and each column of $X$ has $\sqrt{n}$ non-zeros, the matrix $X^J$ acts like a scaled identity matrix).

### Step 3

We can now put everything together. Since $c = x_i$, when we solve $(P1)$ we will get the $i^{th}$ row of $X$ up to scaling. If we solve $(P1)$ repeatedly, we will find all rows of $X$ (and can delete duplicates since now two rows of $X$ will be scaled copies of each other).

Finally we can compute the correct scaling (up to sign) e.g. by using the assumption that non-zero entries are distributed as $\mathcal{N}(0,1)$. Hence we can solve for $X$ up to flipping the sign of its rows, and if $p$ is large enough (i.e. if we take enough samples) then $X$ will have a left pseudo-inverse and we can compute $A = BX^+$ which will recover $A$ up to a permutation and flipping the signs of its columns (which is the best we could hope for).

## 5.3 Overcomplete Dictionaries

Here we will present a recent algorithm of Arora, Ge and Moitra [15] that works for incoherent and overcomplete dictionaries. The crucial idea behind the algorithm is a connection to an overlapping clustering problem. We will consider a model that is similar to that in [108]. Let $k$ be the sparsity parameter:

(a) The support of $x_i$ is chosen uniformly at random from all size $k$ subsets of $[m]$

(b) If the $j$th coordinate is non-zero, then its value is independently chosen to be $+1$ or $-1$ (with equal probability)

The main result in [15] is:

**Theorem 5.3.1** *[15] There is a polynomial time algorithm to learn $A$* exactly *if $A$ is $\mu$-incoherent and $k \leq c\min(\sqrt{n}/\mu \log n, m^{1/2-\eta})$ if we are given samples from the above model. The running time and sample complexity are poly$(n, m)$.*

Recall that methods like K-SVD rely on the intuition that if we have the true dictionary $A$, we can find $X$ and if we have $X$ we can find a good approximation to $A$. However the trouble in analyzing these approaches is that they start from a dictionary that is very far from the true one, so how do these algorithms still make progress? The basic idea of the algorithm in [15] is to break the cycle by first finding the support of $X$ *without knowing the dictionary $A$*.

Our first step is to build a graph which we will call the intersection graph:

**Definition 5.3.2** *Given $p$ samples $b_1, b_2, ..., b_p$ the intersection graph $G$ is a graph on $V = \{b_1, b_2, ..., b_p\}$ where $(b_i, b_j)$ is an edge if and only if $|\langle b_i, b_j \rangle| > \tau$*

How should we choose $\tau$? We want the following properties to hold. Throughout this section we will let $S_i$ denote the support of $x_i$.

(a) If $(i, j)$ is an edge then $S_i \cap S_j \neq \emptyset$

(b) If $S_i \cap S_j \neq \emptyset$ then the probability that $(i, j)$ is an edge is at least $\frac{1}{2}$

Given some row in $X$ we can think of all of the examples where this row is non-zero as belonging to the same *community*. Then the interpretation of the above graph is that if there is an edge between $(i, j)$ we want that $i$ and $j$ necessarily belong to the same community. Moreover if $(i, j)$ do indeed belong to the same community, it should be reasonably likely that there is also an edge.

We can directly compute the above graph given our examples and the basic idea is that we can hope to learn the support of $X$ by finding the communities. The key point is that this departs from standard clustering problems precisely because each sample $x_i$ has $k$ non-zeros and so each node belongs to *polynomially many* communities so what we are looking for is an overlapping clustering.

## Intersection Graph

Here we choose how to set $\tau$ so that the properties above hold with high probability. We can bound
$$|\langle b_i, b_j \rangle - \langle x_i, x_j \rangle| \leq |S_i||S_j|\mu = k^2\mu$$
and hence if $k^2\mu < \frac{1}{2}$ and we choose $\tau = \frac{1}{2}$ we certainly satisfy condition $(a)$ above. Moreover it is not hard to see that if $S_i$ and $S_j$ intersect then the probability that $\langle x_i, x_j \rangle$ is non-zero is at least $\frac{1}{2}$ as well and this yields condition $(b)$. However since $\mu$ is roughly $1/\sqrt{n}$ for random dictionaries (ignoring poly-logarithmic factors), we would need $k < n^{1/4}$ which is a much more stringent condition than we would need to solve the sparse recovery problem.

The above analysis is wasteful in that it does not make use of the random signs of the non-zeros in $X$. We will instead appeal to the following concentration inequality:

**Lemma 5.3.3 (Hanson-Wright)** *Let $x$ be a vector whose components are independent sub-Gaussian random variables, which satisfy $E[X_i] = 0$ and $Var[X_i] = 1$.*

*Let $M$ be a symmetric $n \times n$ matrix. Then, for every $t \geq 0$ we have:*

$$\mathbb{P}[|x^T Mx - trace(M)| > t] \leq 2\ exp\left(-c \min\left(\frac{t^2}{\|M\|_F^2}, \frac{t}{\|M\|_2}\right)\right)$$

Let $S_i$ and $S_j$ be disjoint. Set $N = (A^T A)_{S_j}^{S_i}$ and

$$M = \begin{bmatrix} 0 & \frac{1}{2}N \\ \frac{1}{2}N^T & 0 \end{bmatrix},$$

and let $y$ be the vector which is the concatenation of $x_i$ restricted to $S_i$ and $x_j$ restricted to $S_j$. Then $y^T My = \langle b_i, b_j \rangle$ and we can appeal to the above lemma to bound the deviation of $\langle b_i, b_j \rangle$ from its expectation. In particular, trace$(M) = 0$ and $\|M\|_F^2 \leq \mu^2 k^2$ which implies $\|M\|_2 \leq \mu k$.

Hence if $k \leq \frac{1}{\mu \log n}$ then with high probability we have that $|\langle b_i, b_j \rangle| \leq \frac{1}{3}$. An identical argument works when $S_i$ and $S_j$ intersect (where instead we zero out the entries in $N$ that correspond to the same column in $A$). So if we make use of the randomness in the signs of $X$ the intersection graph (for $\tau = \frac{1}{2}$) satisfies our desired properties provided that $k$ is a $\log n$ factor smaller than what we would need to solve sparse recovery even if we knew $A$.

## Community Finding

Consider the communities $\mathcal{C}_j = \{b_i | S_i \ni j\}$. Then for each pair $b_1, b_2 \in \mathcal{C}_j$ there is an edge $(b_1, b_2)$ with probability at least $\frac{1}{2}$, and moreover our intersection graph can be covered by $m$ dense communities $\{\mathcal{C}_j\}_j$. We will introduce the basic approach through the following thought experiment:

> Suppose we find $b_1, b_2$ and $b_3$ that are a triangle in $G$. We know that $S_1 \cap S_2$, $S_1 \cap S_3$ and $S_2 \cap S_3$ are each non-empty but how do we decide if $S_1 \cap S_2 \cap S_3$ is non-empty too?

Alternatively, given three nodes where each pair belong to the same community, how do we know whether or not all three belong to one community? The intuition is that if $b_1, b_2$ and $b_3$ all belong to a common community then it is more likely that a random new node $b_4$ is connected to all of them than if they don't!

What we need is a lower bound on the probability that $b_4$ connects to each of $b_1, b_2$ and $b_3$ in the case where $S_1 \cap S_2 \cap S_3$ is non-empty, and we need an upper bound when the intersection is empty. It is easy to see that:

**Claim 5.3.4** *If $S_1 \cap S_2 \cap S_3 \neq \emptyset$ then $(b_4, b_i)$ is an edge for $i = 1, 2, 3$ with probability at least $\Omega(k/m)$*

This claim is true because if $i \in S_1 \cap S_2 \cap S_3$, then the probability that $S_4$ contains $i$ is $k/m$. Next we prove the upper bound. Let $a = |S_1 \cap S_2|$, $b = |S_1 \cap S_3|$ and $c = |S_2 \cap S_3|$. Then:

**Lemma 5.3.5** *If $S_1 \cap S_2 \cap S_3 = \emptyset$ the probability that $(b_4, b_i)$ is an edge for $i = 1, 2, 3$ is at most*

$$O\left(\frac{k^6}{m^3} + \frac{k^3(a+b+c)}{m^2}\right)$$

**Proof:** We know that if $(b_4, b_i)$ is an edge for $i = 1, 2, 3$ then $S_4 \cap S_i$ must be non-empty for $i = 1, 2, 3$. We can break up this event into subcases:

(a) Either $S_4$ intersects $S_1$, $S_2$ and $S_3$ disjointly (i.e. it does not contain any index that is in more than one of he other sets)

(b) Or there is some pair $i \neq j \in \{1, 2, 3\}$ so that $S_i \cap S_j$ intersects $S_4$ and $S_4$ intersects the remaining set in another index

The probability of $(a)$ is at most the probability that $|S_4 \cap (S_1 \cup S_2 \cup S_3)| \geq 3$ which is at most $(3k)^3 \frac{k^3}{m^3}$. Similarly the probability of $(b)$ for (say) $i = 1, j = 2$ is at most $\frac{ak^3}{m^2}$. This implies the lemma. ■

We need the lower bound in Claim 5.3.4 to be asymptotically smaller than the upper bound in Lemma 5.3.5. It is easy to see that with high probability for any $i, j$, $|S_i \cap S_j| = O(1)$ and hence we want (roughly)

$$\frac{k}{m} >> \frac{k^6}{m^3} + \frac{k^3}{m^2}$$

which is true if $k < m^{2/5}$. When this holds, for every triple $b_1, b_2, b_3$ we will be able to determine whether or not $S_1 \cap S_2 \cap S_3$ is empty with high probability by counting how many other nodes $b_4$ have an edge to all of them.

Now we are ready to give an algorithm for finding the communities.

```
CommunityFinding [15]
Input: intersection graph $G$
Output: communities $\{\mathcal{C}_j\}$

For each edge $(b_1, b_2)$
    Set $C_{1,2} = \{b_3 | S_1 \cap S_2 \cap S_3 \neq \emptyset\} \cup \{b_1, b_2\}$
End
Remove sets $C_{1,2}$ that strictly contain another set $C_{i,j}$
```

**Definition 5.3.6** *If $S_1 \cap S_2 = \{j\}$ we will call the pair $(b_1, b_2)$ an identifying pair for community $j$*

It is easy to see that with high probability each community will have an identifying pair if the number of samples $p$ is large enough. Then consider

$$C_{1,2} = \{b_3 | S_1 \cap S_2 \cap S_3 \neq \emptyset\} \cup \{b_1, b_2\}$$

If $(b_1, b_2)$ is an identifying pair for community $j$ then the above set $C_{1,2}$ is exactly $\mathcal{C}_j$ (and we can compute this set by using the above test for deciding whether or not $S_1 \cap S_2 \cap S_3$ is empty).

Moreover if $(b_1, b_2)$ is not an identifying pair but rather $S_1 \cap S_2$ has more than one element we would have instead $C_{1,2} = \cup_{j \in S_1 \cap S_2} \mathcal{C}_j$ in which case the set $C_{1,2}$ will be deleted in the last step. This algorithm outputs the correct communities with high probability if $k \leq c \min(\sqrt{n}/\mu \log n, m^{2/5})$. In [15] the authors give a higher-order algorithm for community finding that works when $k \leq m^{1/2-\eta}$ for any $\eta > 0$, however the running time is a polynomial whose exponent depends on $\eta$.

All that remains to recover the true dictionary is to partition the community $\mathcal{C}_j$ into those where the $j$th coordinate is $+1$ and those where it is $-1$. In fact if $S_1 \cap S_2 = \{j\}$ then the sign of $\langle b_1, b_2 \rangle$ tells us whether or not their $j$th coordinate has the same sign or a different sign. It is not hard to show that there are enough such pairs in a typical community that we can successfully partition $\mathcal{C}_j$ into two sets where all the examples in one have their $j$th coordinate equal to $+1$ and all those in the other are $-1$ (of course we do not know which is which). This in turn allows us to compute $X$ up to a permutation or flipping the signs of its rows, and again we can set $A = BX^+$ and compute the true dictionary exactly.

# Chapter 6

# Gaussian Mixture Models

In this chapter we will study *Gaussian mixture models* and clustering. The basic problem is, given random samples from a mixture of $k$ Gaussians, we would like to give an efficient algorithm to learn its parameters using few samples. If these parameters are accurate, we can then cluster the samples and our error will be nearly as accurate as the Bayes optimal classifier.

## 6.1 History

The problem of learning the parameters of a mixture of Gaussians dates back to the famous statistician Karl Pearson (1894) who was interested in biology and evolution. In fact, there was a particular species of crab called the Naples crab that inhabited the region around him. He took thousands of samples from this population and measured some physical characteristic of each sample. He plotted the frequency of occurrence, but the resulting density function surprised him. He expected that it would be Gaussian, but in fact it was not even symmetric around its maximum value. See Figure 6.1. He hypothesized that maybe the Naples crab was not one species but rather two, and that the density function he observed could be explained as a mixture of Gaussians.

In this remarkable study Pearson introduced the *method of moments*. His basic idea was to compute empirical moments from his samples, and use each of these empirical moments to set up a system of polynomial equations on the parameters of the mixture. He solved this system *by hand*! In fact, we will return to his basic approach later in this unit.

## Basics

Here we formally describe the problem of learning mixtures of Gaussians. Recall that for a univariate Gaussian we have that its density function is given by:

$$\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left\{\frac{-(x-\mu)^2}{2\sigma^2}\right\}$$

The density of a multidimensional Gaussian in $\mathbb{R}^n$ is given by:

$$\mathcal{N}(\mu, \Sigma) = \frac{1}{(2\pi)^{n/2} det(\Sigma)^{1/2}} exp\left\{\frac{-(x-\mu)^\top \Sigma^{-1}(x-\mu)}{2}\right\}$$

Here $\Sigma$ is the covariance matrix. If $\Sigma = I_n$ and $\mu = \vec{0}$ then the distribution is just: $\mathcal{N}(0,1) \times \mathcal{N}(0,1) \times ... \times \mathcal{N}(0,1)$.

A mixture of two Gaussians is a distribution whose density function is:

$$F(x) = w_1 F_1(x) + (1 - w_1) F_2(x)$$

where $F_1$ and $F_2$ are Gaussians. We can generate a random sample as follows: with probability $w_1$ we output a random sample from $F_1$, and otherwise we output a random sample from $F_2$. Our basic problem is to learn the parameters that describe the mixture given random samples from $F$. We note that we will measure how good an algorithm is by both its *sample complexity* and its running time.

## Method of Moments

Pearson used the *method of moments* to fit a mixture of two Gaussians to his data. The moments of a mixture of Gaussians are themselves a polynomial in the unknown parameters, which we will denote by $M_r$.

$$\mathop{\mathbb{E}}_{x \leftarrow F_1(x)}[x^r] = M_r(\mu, \sigma^2)$$

Then we can write

$$\mathop{\mathbb{E}}_{x \leftarrow F(x)}[x^r] = w_1 M_r(\mu_1, \sigma_1^2) + (1 - w_1) M_r(\mu_2, \sigma_2^2) = P_r(w_1, \mu_1, \sigma_1, \mu_2^2, \sigma_2^2)$$

And hence the $r^{th}$ raw moment of a mixture of two Gaussians is itself a degree $r+1$ polynomial ($P_r$) in the unknown parameters.

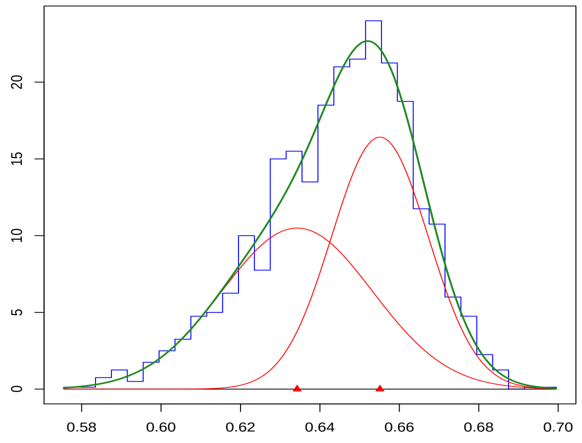

Figure 6.1: A fit of a mixture of two univariate Gaussians to the Pearson's data on Naples crabs, created by Peter Macdonald using R

**Pearson's Sixth Moment Test:** We can estimate $\mathbb{E}_{x\leftarrow F}[x^r]$ from random samples: Let $S$ be our set of samples. Then we can compute:

$$\widetilde{M}_r = \frac{1}{|S|}\sum_{x\in S} x^r$$

And given a polynomial number of samples (for any $r = O(1)$), $\widetilde{M}_r$ will be additively close to $\mathbb{E}_{x\leftarrow F(x)}[x^r]$. Pearson's approach was:

- Set up a system of polynomial equations
$$\left\{P_r(w_1, \mu_1, \sigma_1, \mu_2^2, \sigma_2^2) = \widetilde{M}_r\right\},\ r = 1, 2, ...5$$
- Solve this system. Each solution is a setting of all five parameters that explains the first five empirical moments.

Pearson solved the above system of polynomial equations *by hand*, and he found a number of candidate solutions. Each solution corresponds to a simultaneous setting of the parameters so that the moments of the mixture would match the empirical moments. But how can we choose among these candidate solutions? Some of the solutions were clearly not right; some had negative values for the variance, or a value for the mixing weight not in the range $[0, 1]$. But even after eliminating these solutions, Pearson was still left with more than one candidate. His approach was to choose the root whose prediction is closest to the empirical sixth moment $\widetilde{M}_6$. This is called the *sixth moment test*.

### Expectation-Maximization

Much of modern statistics instead focuses on the *maximum likelihood estimator*, which would choose to set the parameters to as to maximize the probability that the mixture would generate the observed samples. Unfortunately, this estimator is $NP$-hard to compute [18]. The popular alternative is known as *expectation-maximization* and was introduced in a deeply influential paper of Dempster, Laird, Rubin [50]. The basic approach is to repeat the following steps until convergence:

- For each $x \in S$, calculate the posterior probability:

$$w_1(x) = \frac{w_1 F_1(x)}{w_1 F_1(x) + (1 - w_1) F_2(x)}$$

- Update the mixing weights:

$$w_1 \leftarrow \frac{\sum_{x \in S} w_1(x)}{|S|}$$

- Re-estimate the parameters:

$$\mu_i \leftarrow \frac{\sum_{x \in S} w_i(x) x}{\sum_{x \in S} w_i(x)}, \ \Sigma_i \leftarrow \frac{\sum_{x \in S} w_i(x)(x - \mu_i)(x - \mu_i)^\top}{\sum_{x \in S} w_i(x)}$$

This approach gets stuck in local optima, and is in general quite sensitive to how it is initialized (see e.g. [105]).

## 6.2 Clustering-Based Algorithms

Our basic goal will be to give algorithms that provably compute the true parameters of a mixture of Gaussians, given a polynomial number of random samples. This question was introduced in the seminal paper of Dasgupta [45], and the first generation of algorithms focused on the case where the components of the mixture have essentially no "overlap". The next generation algorithms are based on algebraic ideas, and avoid clustering altogether.

Before we proceed, we will discuss some of the counter-intuitive properties of high-dimensional Gaussians. To simplify the discussion, we will focus on spherical Gaussians $\mathcal{N}(\mu, \sigma^2 I)$ in $\mathbb{R}^n$.

**Fact 6.2.1** *The maximum value of the density function is at $x = \mu$.*

**Fact 6.2.2** *Almost all of the weight of the density function has* $\|x-\mu\|_2^2 = \sigma^2 n \pm \sigma^2\sqrt{n \log n}$

These facts seem to be inconsistent, but the explanation is that the surface area increases faster as the radius $R$ increases than the value of the density function decreases, until we reach $R^2 \approx \sigma^2 n$. Hence we should think about a high-dimensional spherical Gaussian as being a ball of radius $\sigma\sqrt{n}$ with a thin shell.

### Dasgupta [45] – $\widetilde{\Omega}(\sqrt{n})$ Separation

Dasgupta gave the first provable algorithms for learning mixtures of Gaussians, and required that $\|\mu_i - \mu_j\|_2 \geq \widetilde{\Omega}(\sqrt{n}\sigma_{max})$ where $\sigma_{max}$ is the maximum variance of any Gaussian in any direction (e.g. if the components are not spherical). Note that the constant in the separation depends on $w_{min}$, and we assume we know this parameter (or a lower bound on it).

The basic idea behind the algorithm is to project the mixture onto $\log k$ dimensions uniformly at random. This projection will preserve distances between each pair of centers $\mu_i$ and $\mu_j$ with high probability, but will contract distances between samples from the same component and make each component closer to spherical, thus making it easier to cluster. We can then cluster all of the samples into which component generated them, and then for each cluster we can choose the empirical mean and empirical covariance which will with high probability be a good estimate of $\mu_i$ and $\Sigma_i$. Additionally we can estimate $w_i$ by how large each cluster is.

Informally, we can think of this separation condition as: if we think of each Gaussian as a spherical ball, then if the components are far enough apart then these balls will be *disjoint*.

### Arora and Kannan [18], Dasgupta and Schulman [53] – $\widetilde{\Omega}(n^{1/4})$ Separation

We will describe the approach in [18] in detail. The basic question is, if $\sqrt{n}$ separation is the threshold when we can think of the components as disjoint, then how can we learn when the components are much closer? In fact, even if the components are only $\widetilde{\Omega}(n^{1/4})$ separated then it is still true that *every* pair of samples from the same component is closer than *every* pair of samples from different components. How can this be? The explanation is that even though the balls representing each component are no longer disjoint, we are still very unlikely to sample from their overlap region.

Consider $x, x' \leftarrow F_1$ and $y \leftarrow F_2$.

**Claim 6.2.3** *All of the vectors $x-\mu_1$, $x'-\mu_1$, $\mu_1-\mu_2$, $y-\mu_2$ are nearly orthogonal (whp)*

This claim is immediate since the vectors $x-\mu_1$, $x'-\mu_1$, $y-\mu_2$ are uniform from a sphere, and $\mu_1-\mu_2$ is the only fixed vector. In fact, any set of vectors in which all but one is uniformly random from a sphere are nearly orthogonal.

Now we can compute:

$$\begin{aligned}\|x-x'\|^2 &\approx \|x-\mu_1\|^2+\|\mu_1-x'\|^2\\ &\approx 2n\sigma^2 \pm 2\sigma^2\sqrt{n\log n}\end{aligned}$$

And similarly:

$$\begin{aligned}\|x-y\|^2 &\approx \|x-\mu_1\|^2+\|\mu_1-\mu_2\|^2+\|\mu_2-y\|^2\\ &\approx 2n\sigma^2+\|\mu_1-\mu_2\|^2 \pm 2\sigma^2\sqrt{n\log n}\end{aligned}$$

Hence if $\|\mu_1-\mu_2\| = \widetilde{\Omega}(n^{1/4},\sigma)$ then $\|\mu_1-\mu_2\|^2$ is larger than the error term and each pair of samples from the same component will be closer than each pair from different components. Indeed we can find the right threshold $\tau$ and correctly cluster all of the samples. Again, we can output the empirical mean, empirical covariance and relative size of each cluster and these will be good estimates of the true parameters.

### Vempala and Wang [117] – $\widetilde{\Omega}(k^{1/4})$ Separation

Vempala and Wang [117] removed the dependence on $n$, and replaced it with a separation condition that depends on $k$ – the number of components. The idea is that if we could project the mixture into the subspace $T$ spanned by $\{\mu_1,\dots,\mu_k\}$, we would preserve the separation between each pair of components but reduce the ambient dimension.

So how can we find $T$, the subspace spanned by the means? We will restrict our discussion to a mixture of spherical Gaussians with a common variance $\sigma^2 I$. Let $x\sim F$ be a random sample from the mixture, then we can write $x=c+z$ where $z\sim N(0,\sigma^2 I_n)$ and $c$ is a random vector that takes the value $\mu_i$ with probability $w_i$ for each $i\in[k]$. So:

$$\mathbb{E}[xx^T]=E[cc^T]+E[zz^T]=\sum_{i=1}^{k} w_i\mu_i\mu_i^\top+\sigma^2 I_n$$

Hence the top left singular vectors of $\mathbb{E}[xx^T]$ whose singular value is strictly larger than $\sigma^2$ exactly span $T$. We can then estimate $\mathbb{E}[xx^T]$ from sufficiently many random samples, compute its singular value decomposition and project the mixture onto $T$ and invoke the algorithm of [18].

**Brubaker and Vempala [32] – Separating Hyperplane**

What if the largest variance of any component is much larger than the separation between the components? Brubaker and Vempala [32] observed that none of the existing algorithms succeed for the *parallel pancakes* example, depicted in Figure **??** even though there is a hyperplane that separates the mixture so that almost all of one component is on one side, and almost all of the other component is on the other side. [32] gave an algorithm that succeeds, provided there is such a separating hyperplane, however the conditions are more complex to state for mixtures of more than two Gaussians. Note that not all mixtures that we could hope to learn have such a separating hyperplane. See e.g. Figure **??**.

## 6.3 Discussion of Density Estimation

The algorithms we have discussed so far [45], [53], [18], [117], [1], [32] have focused on clustering; can we give efficient learning algorithms even when clustering is *impossible*? Consider a mixture of two Gaussians $F = w_1 F_1 + w_2 F_2$. The separation conditions we have considered so far each imply that $d_{TV}(F_1, F_2) = 1 - o(1)$. In particular, the components have negligible overlap. However if $d_{TV}(F_1, F_2) = 1/2$ we cannot hope to learn which component generated each sample.

More precisely, the total variation distance between two distributions $F$ and $G$ measures how well we can couple them:

**Definition 6.3.1** *A coupling between $F$ and $G$ is a distribution on pairs $(x, y)$ so that the marginal distribution on $x$ is $F$ and the marginal distribution on $y$ is $G$. The error is the probability that $x \neq y$.*

**Claim 6.3.2** *There is a coupling with error $\varepsilon$ between $F$ and $G$ if and only if $d_{TV}(F, G) \leq \varepsilon$.*

Returning to the problem of clustering the samples from a mixture of two Gaussians, we have that if $d_{TV}(F_1, F_2) = 1/2$ then there is a coupling between $F_1$ and $F_2$ that agrees with probability 1/2. Hence instead of thinking about sampling from a mixture of two Gaussians in the usual way (choose which component, then choose a random sample from it) we can alternatively sample as follows:

(a) Choose $(x, y)$ from the best coupling between $F_1$ and $F_2$

(b) If $x = y$, output $x$

(c) Else output $x$ with probability $w_1$, and otherwise output $y$

This procedure generates a random sample from $F$, but for half of the samples we did not need to decide which component generated it at all! Hence *even if we knew the mixture* there is no clustering procedure that can correctly classify a polynomial number of samples into which component generated them! So in the setting where $d_{TV}(F_1, F_2)$ is not $1 - o(1)$, the fundamental approach we have discussed so far does not work! Nevertheless we will be able to give algorithms to learn the parameters of $F$ even when $d_{TV}(F_1, F_2) = o(1)$ and the components almost entirely overlap.

Next we will discuss some of the basic types of goals for learning algorithms:

(a) **Improper Density Estimation**

Throughout we will assume that $F \in \mathcal{C}$ where $\mathcal{C}$ is some class of distributions (e.g. mixtures of two Gaussians). Our goal in improper density estimation is to find any distribution $\widehat{F}$ so that $d_{TV}(F, \widehat{F}) \leq \varepsilon$. This is the weakest goal for a learning algorithm. A popular approach (especially in low dimension) is to construct a *kernel density estimate*; suppose we take many samples from $F$ and construct a point-mass distribution $G$ that represents our samples. Then we can set $\widehat{F} = G * \mathcal{N}(0, \sigma^2)$, and if $F$ is smooth enough and we take enough samples, $d_{TV}(F, \widehat{F}) \leq \varepsilon$. However $\widehat{F}$ works without learning anything about the components of $F$; it works just because $F$ is smooth. We remark that such an approach fails badly in high dimensions where even if $F$ is smooth, we would need to take an exponential number of samples in order to guarantee that $\widehat{F} = G * \mathcal{N}(0, \sigma^2 I)$ is close to $F$.

(b) **Proper Density Estimation**

Here, our goal is to find a distribution $\widehat{F} \in \mathcal{C}$ where $d_{TV}(F, \widehat{F}) \leq \varepsilon$. Note that if $\mathcal{C}$ is the set of mixtures of two Gaussians, then a kernel density estimate is not a valid hypothesis since it will in general be a mixture of many Gaussians (as many samples as we take). Proper density estimation is in general much harder to do than improper density estimation. In fact, we will focus on an even stronger goal:

(b) **Parameter Learning**

Here we require not only that $d_{TV}(F, \widehat{F}) \leq \varepsilon$ and that $\widehat{F} \in \mathcal{C}$, but we want $\widehat{F}$ to be a good estimate for $F$ *on a component-by-component basis*. For example, our goal specialized to the case of mixtures of two Gaussians is:

**Definition 6.3.3** *We will say that a mixture $\widehat{F} = \widehat{w}_1\widehat{F}_1 + \widehat{w}_2\widehat{F}_2$ is $\varepsilon$-close (on a component-by-component basis) to $F$ if there is a permutation $\pi : \{1,2\} \to \{1,2\}$ so that for all $i \in \{1,2\}$:*

$$\left|w_i - \widehat{w}_{\pi(i)}\right|, d_{TV}(F_i, \widehat{F}_{\pi(i)}) \leq \varepsilon$$

Note that $F$ and $\widehat{F}$ must necessarily be close as mixtures too: $d_{TV}(F, \widehat{F}) \leq 4\varepsilon$. However we can have mixtures $F$ and $\widehat{F}$ that are both mixtures of $k$ Gaussians, are close as distributions but are not close on a component-by-component basis. It is better to learn $F$ on a component-by-component basis than to do only proper density estimation, if we can. Note that if $\widehat{F}$ is $\varepsilon$-close to $F$, then even when we cannot cluster samples we will still be able to approximately compute the posterior [79] and this is one of the main advantages of parameter learning over some of the weaker learning goals.

But one should keep in mind that *lower bounds for parameter learning do not imply lower bounds for proper density estimation.* We will give optimal algorithms for parameter learning for mixtures of $k$ Gaussians, which run in polynomial time for any $k = O(1)$. Moreover there are pairs of mixtures of $k$ Gaussians $F$ and $\widehat{F}$ that are not close on a component-by-component basis, but have $d_{TV}(F, \widehat{F}) \leq 2^{-k}$ [95]. Hence there is no algorithm for parameter learning that takes $\text{poly}(n, k, 1/\varepsilon)$ samples – because we need to take at least $2^k$ samples to distinguish $F$ and $\widehat{F}$. But in the context of proper density estimation, we do not need to distinguish these two mixtures.

**Open Question 2** *Is there a* $\text{poly}(n, k, 1/\varepsilon)$ *time algorithm for proper density estimation for mixtures of $k$ Gaussians in $n$ dimensions?*

## 6.4 Clustering-Free Algorithms

Recall, our goal is to learn $\widehat{F}$ that is $\varepsilon$-close to $F$. In fact, the same definition can be generalized to mixtures of $k$ Gaussians:

**Definition 6.4.1** *We will say that a mixture $\widehat{F} = \sum_{i=1}^{k} \widehat{w}_i\widehat{F}_i$ is $\varepsilon$-close (on a component-by-component basis) to $F$ if there is a permutation $\pi : \{1,2,...,k\} \to \{1,2,...,k\}$ so that for all $i \in \{1,2,...,k\}$:*

$$\left|w_i - \widehat{w}_{\pi(i)}\right|, d_{TV}(F_i, \widehat{F}_{\pi(i)}) \leq \varepsilon$$

When can we hope to learn an $\varepsilon$ close estimate in $\mathrm{poly}(n, 1/\varepsilon)$ samples? In fact, there are two trivial cases where we cannot do this, but these will be the only things that go wrong:

(a) If $w_i = 0$, we can never learn $\widehat{F}_i$ that is close to $F_i$ because we never get any samples from $F_i$.

In fact, we need a quantitative lower bound on each $w_i$, say $w_i \geq \varepsilon$ so that if we take a reasonable number of samples we will get at least one sample from each component.

(b) If $d_{TV}(F_i, F_j) = 0$ we can never learn $w_i$ or $w_j$ because $F_i$ and $F_j$ entirely overlap.

Again, we need a quantitive lower bound on $d_{TV}(F_i, F_j)$, say $d_{TV}(F_i, F_j) \geq \varepsilon$ for each $i \neq j$ so that if we take a reasonable number of samples we will get at least one sample from the non-overlap region between various pairs of components.

**Theorem 6.4.2** *[79], [95] If $w_i \geq \varepsilon$ for each $i$ and $d_{TV}(F_i, F_j) \geq \varepsilon$ for each $i \neq j$, then there is an efficient algorithm that learns an $\varepsilon$-close estimate $\widehat{F}$ to $F$ whose running time and sample complexity are* $\mathrm{poly}(n, 1/\varepsilon, \log 1/\delta)$ *and succeeds with probability* $1 - \delta$.

Note that the degree of the polynomial depends polynomially on $k$. Kalai, Moitra and Valiant [79] gave the first algorithm for learning mixtures of two Gaussians with no separation conditions. Subsequently Moitra and Valiant [95] gave an algorithm for mixtures of $k$ Gaussians, again with no separation conditions.

In independent and concurrent work, Belkin and Sinha [23] gave a polynomial time algorithm for mixtures of $k$ Gaussians too, however there is no explicit bound given on the running time as a function of $k$ (since their work depends on the basis theorem, which is provably ineffective). Also, the goal in [79] and [95] is to learn $\widehat{F}$ so that its components are close in total variation distance to those of $F$, which is in general a stronger goal than requiring that the parameters be additively close which is the goal in [23]. The benefit is that the algorithm in [23] works for more general learning problems in the one-dimensional setting, and we will describe this algorithm in detail at the end of this chapter.

Throughout this section, we will focus on the $k = 2$ case since this algorithm is conceptually much simpler. In fact, we will focus on a weaker learning goal: We will say that $\widehat{F}$ is *additively* $\varepsilon$-close to $F$ if $|w_i - \widehat{w}_{\pi(i)}|, \|\mu_i - \widehat{\mu}_{\pi(i)}\|, \|\Sigma_i - \widehat{\Sigma}_{\pi(i)}\|_F \leq \varepsilon$ for all $i$. We will further assume that $F$ is normalized appropriately:

**Definition 6.4.3** *A distribution $F$ is in isotropic position if*

*(a)* $\mathbb{E}_{x\leftarrow F}[x] = 0$

*(b)* $\mathbb{E}_{x\leftarrow F}[xx^T] = I$

Alternatively, we require that the mean of the distribution is zero and that its variance *in every direction* is one. In fact this condition is not quite so strong as it sounds:

**Claim 6.4.4** *If $\mathbb{E}_{x\leftarrow F}[xx^T]$ is full-rank, then there is an affine transformation that places $F$ in isotropic position*

**Proof:** Let $\mu = E_{x\leftarrow F}[x]$ and let $E_{x\leftarrow F}[(x-\mu)(x-\mu)^T] = M$. It is easy to see that $M$ is positive semi-definite, and in fact is full rank by assumption. Hence we can write $M = BB^T$ where $B$ is invertible (this is often referred to as the Cholesky decomposition [74]). Then set $y = B^{-1}(x-\mu)$, and it is easy to see that $\mathbb{E}[y] = 0$ and $\mathbb{E}[yy^T] = B^{-1}M(B^{-1})^T = I$. ■

Our goal is to learn an additive $\varepsilon$ approximation to $F$, and we will assume that $F$ has been pre-processed so that it is in isotropic position.

### Outline

We can now describe the basic outline of the algorithm, although there will be many details to fill:

(a) Consider a series of projections down to one dimension

(b) Run a univariate learning algorithm

(c) Set up a system of linear equations on the high-dimensional parameters, and back solve

## Isotropic Projection Lemma

We will need to overcome a number of obstacles to realize this plan, but let us work through the details of this outline:

**Claim 6.4.5** $proj_r[\mathcal{N}(\mu,\Sigma)] = \mathcal{N}(r^T\mu, r^T\Sigma r)$

Alternatively, the projection of a high-dimensional Gaussian is a one-dimensional Gaussian, and its mean and variance are $r^T\mu$ and $r^T\Sigma r$ respectively. This implies that if we knew the parameters of the projection of a single Gaussian component onto a (known) direction $r$, then we could use these parameters to set up a linear constraint for $\mu$ and $\Sigma$. If we follow this plan, we would need to consider about $n^2$ projections to get enough linear constraints, since there are $\Theta(n^2)$ variances in $\Sigma$ that we need to solve for. Now we will encounter the first problem in the outline. Let us define some notation:

**Definition 6.4.6** $d_p(\mathcal{N}(\mu_1,\sigma_1^2),\mathcal{N}(\mu_2,\sigma_2^2)) = |\mu_1-\mu_2| + |\sigma_1^2-\sigma_2^2|$

We will refer to this as the parameter distance. Ultimately, we will give a univariate algorithm for learning mixtures of Gaussians and we would like to run it on $\text{proj}_r[F]$.

**Problem 4** *But what if $d_p(proj_r[F_1], proj_r[F_2])$ is exponentially small?*

This would be a problem since we would need to run our univariate algorithm with exponentially fine precision just to see that there are two components and not one! How can we get around this issue? In fact, this almost surely never happens provided that $F$ is in isotropic position. For intuition, consider two cases:

(a) Suppose $\|\mu_1-\mu_2\| \geq \text{poly}(1/n,\varepsilon)$.

If the difference between the means of $F_1$ and $F_2$ is at least any fixed inverse polynomial, then with high probability $\|r^T\mu_1 - r^T\mu_2\|$ is at least $\text{poly}(1/n,\varepsilon)$ too. Hence $\text{proj}_r[F_1]$ and $\text{proj}_r[F_2]$ will have different parameters due to a difference in their means.

(b) Suppose $\|\mu_1-\mu_2\| \leq \text{poly}(1/n,\varepsilon)$.

The key observation is that if $d_{TV}(F_1,F_2)\geq\varepsilon$ and their means are almost identical, then their covariances $\Sigma_1$ and $\Sigma_2$ must be noticeably different when projected on a random direction $r$. In this case, $\text{proj}_r[F_1]$ and $\text{proj}_r[F_2]$ will have different parameters due to a difference in their variances. This is the intuition behind the following lemma:

**Lemma 6.4.7** *If $F$ is in isotropic position and $w_i\geq\varepsilon$ and $d_{TV}(F_1,F_2)\geq\varepsilon$, then with high probability for a random $r$*

$$d_p(proj_r[F_1], proj_r[F_2]) \geq 2\varepsilon_3 = \text{poly}(1/n,\varepsilon)$$

Note that this lemma is note true when $F$ is not in isotropic position (e.g. consider the parallel pancakes example), and moreover when generalizing to mixtures of $k > 2$ Gaussians this is the key step that fails since even if $F$ is in isotropic position, it could be that for almost all choices of $r$ the projection onto $r$ results in a mixtures that is exponentially closet to a mixture of $< k$ Gaussians! (The approach in [95] is to learn a mixture of $< k$ Gaussians as a proxy for the true mixture, and later on find a direction that can be used to cluster the mixture into sub mixtures and recurse).

## Pairing Lemma

Next we will encounter the second problem: Suppose we project onto direction $r$ and $s$ and learn $\widehat{F}^r = \frac{1}{2}\widehat{F}_1^r + \frac{1}{2}\widehat{F}_2^r$ and $\widehat{F}^s = \frac{1}{2}\widehat{F}_1^s + \frac{1}{2}\widehat{F}_2^s$ respectively. Then the mean and variance of $\widehat{F}_1^r$ yield a linear constraint on one of the two high-dimensional Gaussians, and similarly for $\widehat{F}_1^s$.

**Problem 5** *How do we know that they yield constraints on the* same *high-dimensional component?*

Ultimately we want to set up a system of linear constraints to solve for the parameters of $F_1$, but when we project $F$ onto different directions (say, $r$ and $s$) we need to pair up the components from these two directions. The key observation is that as we vary $r$ to $s$ the parameters of the mixture vary continuously. See Figure **??**. Hence when we project onto $r$, we know from the isotropic projection lemma that the two components will either have noticeably different means or variances. Suppose their means are different by $\varepsilon_3$; then if $r$ and $s$ are close (compared to $\varepsilon_1$) the parameters of each component in the mixture do not change much and the component in $\text{proj}_r[F]$ with larger mean will correspond to the same component as the one in $\text{proj}_s[F]$ with larger mean. A similar statement applies when it is the variances that are at least $\varepsilon_3$ apart.

**Lemma 6.4.8** *If* $\|r - s\| \le \varepsilon_2 = \text{poly}(1/n, \varepsilon_3)$ *then*

*(a) If* $|r^T\mu_1 - r^T\mu_2| \ge \varepsilon_3$ *then the components in* $\text{proj}_r[F]$ *and* $\text{proj}_s[F]$ *with the larger mean correspond to the same high-dimensional component*

*(b) Else if* $|r^T\Sigma_1 r - r^T\Sigma_2 r| \ge \varepsilon_3$ *then the components in* $\text{proj}_r[F]$ *and* $\text{proj}_s[F]$ *with the larger variance correspond to the same high-dimensional component*

Hence if we choose $r$ randomly and only search over directions $s$ with $\|r - s\| \le \varepsilon_2$, we will be able to pair up the components correctly in the different one-dimensional mixtures.

### Condition Number Lemma

Now we encounter the final problem in the high-dimensional case: Suppose we choose $r$ randomly and for $s_1, s_2, ...., s_p$ we learn the parameters of the projection of $F$ onto these directions and pair up the components correctly. We can only hope to learn the parameters on these projection up to some additive accuracy $\varepsilon_1$ (and our univariate learning algorithm will have running time and sample complexity $\text{poly}(1/\varepsilon_1)$).

**Problem 6** *How do these errors in our univariate estimates translate to errors in our high dimensional estimates for $\mu_1, \Sigma_1, \mu_2, \Sigma_2$?*

Recall that the *condition number* controls this. The final lemma we need in the high-dimensional case is:

**Lemma 6.4.9** *The condition number of the linear system to solve for $\mu_1, \Sigma_1$ is* $\text{poly}(1/\varepsilon_2, n)$ *where all pairs of directions are $\varepsilon_2$ apart.*

Intuitively, as $r$ and $s_1, s_2, ...., s_p$ are closer together then the condition number of the system will be worse (because the linear constraints are closer to redundant), but the key fact is that the condition number is bounded by a fixed polynomial in $1/\varepsilon_2$ and $n$, and hence if we choose $\varepsilon_1 = \text{poly}(\varepsilon_2, n)\varepsilon$ then our estimates to the high-dimensional parameters will be within an additive $\varepsilon$. Note that each parameter $\varepsilon, \varepsilon_3, \varepsilon_2, \varepsilon_1$ is a fixed polynomial in the earlier parameters (and $1/n$) and hence we need only run our univariate learning algorithm with inverse polynomial precision on a polynomial number of mixtures to learn an $\varepsilon$-close estimate $\widehat{F}$!

But we still need to design a univariate algorithm, and next we return to Pearson's original problem!

## 6.5 A Univariate Algorithm

Here we will give a univariate algorithm to learning the parameters of a mixture of two Gaussians up to additive accuracy $\varepsilon$ whose running time and sample complexity is $\text{poly}(1/\varepsilon)$. Note that the mixture $F = w_1F_1 + w_2F_2$ is in isotropic position (since the projection of a distribution in isotropic position is itself in isotropic position), and as before we assume $w_1, w_2 \geq \varepsilon$ and $d_{TV}(F_1, F_2) \geq \varepsilon$. Our first observation is that all of the parameters are bounded:

**Claim 6.5.1** *(a)* $\mu_1, \mu_2 \in [-1/\sqrt{\varepsilon}, 1/\sqrt{\varepsilon}]$

*(b)* $\sigma_1^2, \sigma_2^2 \in [0, 1/\varepsilon]$

This claim is immediate, since if any of the above conditions are violated it would imply that the mixture has variance strictly larger than one (because $w_1, w_2 \geq \varepsilon$ and the mean of the mixture is zero).

Hence we could try to learn the parameters using a *grid search*:

```
Grid Search
Input: samples from $F(\Theta)$
Output: parameters $\widehat{\Theta} = (\widehat{w}_1, \widehat{\mu}_1, \widehat{\sigma}_1^2, \widehat{\mu}_2, \widehat{\sigma}_2^2)$

For all valid $\widehat{\Theta}$ where the parameters are multiples of $\varepsilon^C$
   Test $\widehat{\Theta}$ using the samples, if it passes output $\widehat{\Theta}$
End
```

For example, we could test out $\widehat{\Theta}$ by computing the first six moments of $F(\Theta)$ from enough random examples, and output $\widehat{\Theta}$ if its first six moments are each within an additive $\tau$ of the observed moments. (This is a slight variant on Pearson's sixth moment test).

It is easy to see that if we take enough samples and set $\tau$ appropriately, then if we round the true parameters $\Theta$ to any valid grid point whose parameters are multiples of $\varepsilon^C$, then the resulting $\widehat{\Theta}$ will with high probability pass our test. This is called the *completeness*. The much more challenging part is establishing the *soundness*; after all why is there no other set of parameters $\widehat{\Theta}$ except for ones close to $\Theta$ that pass our test?

Alternatively, we want to prove that any two mixtures $F$ and $\widehat{F}$ whose parameters *do not* match within an additive $\varepsilon$ must have one of their first six moments noticeably different. The main lemma is:

**Lemma 6.5.2** *For any $F$ and $\widehat{F}$ that are not $\varepsilon$-close in parameters, there is an $r \in \{1, 2, ..., 6\}$ where*

$$\left| M_r(\Theta) - M_r(\widehat{\Theta}) \right| \geq \varepsilon^{O(1)}$$

*where $\Theta$ and $\widehat{\Theta}$ are the parameters of $F$ and $\widehat{F}$ respectively, and $M_r$ is the $r^{th}$ raw moment.*

Let $\widetilde{M_r}$ be the empirical moments. Then

$$\left| M_r(\widehat{\Theta}) - M_r(\Theta) \right| \leq \underbrace{\left| \widetilde{M_r}(\widehat{\Theta}) - \widetilde{M_r} \right|}_{\leq \tau} + \underbrace{\left| \widetilde{M_r} - M_r(\Theta) \right|}_{\leq \tau} \leq 2\tau$$

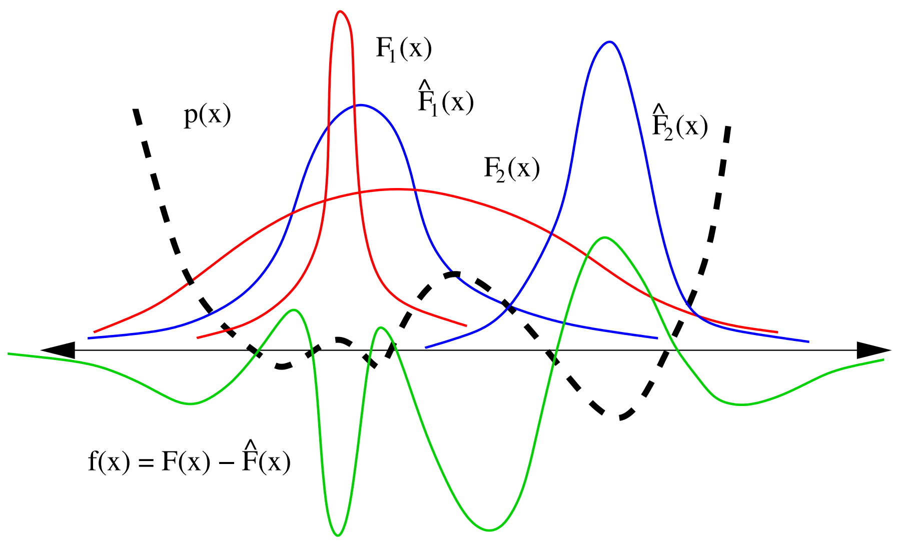

Figure 6.2: If $f(x)$ has at most six zero crossings, we can find at most degree six polynomial that agrees with its sign

where the first term is at most $\tau$ because the test passes and the second term is small because we can take enough samples (but still poly$(1/\tau)$) so that the empirical moments and the true moments are close. Hence we can apply the above lemma in the contrapositive, and conclude that if the grid search outputs $\widehat{\Theta}$ then $\Theta$ and $\widehat{\Theta}$ must be $\varepsilon$-close in parameters, which gives us an efficient univariate algorithm!

So our main goal is to prove that if $F$ and $\widehat{F}$ that are not $\varepsilon$-close, then one of their first six moments is noticeably different. In fact, even the case of $\varepsilon = 0$ is challenging: If $F$ and $\widehat{F}$ are different mixtures of two Gaussians, why is one of their first six moments necessarily different? Our main goal is to prove this statement, using the *heat equation*.

In fact, let us consider the following thought experiment. Let $f(x) = F(x) - \widehat{F}(x)$ be the point-wise difference between the density functions $F$ and $\widehat{F}$. Then, the heart of the problem is: Can we prove that $f(x)$ crosses the $x$-axis at most six times? See Figure 6.2.

**Lemma 6.5.3** *If $f(x)$ crosses the $x$-axis at most six times, then one of the first six moments of $F$ and $\widehat{F}$ are different*

**Proof:** In fact, we can construct a (non-zero) degree at most six polynomial $p(x)$ that agrees with the sign of $f(x)$ – i.e. $p(x)f(x) \geq 0$ for all $x$. Then

$$\begin{aligned} 0 < \Big| \int_x p(x)f(x)dx \Big| &= \Big| \int_x \sum_{r=1}^{6} p_r x^r f(x)dx \Big| \\ &\leq \sum_{r=1}^{6} |p_r| \Big| M_r(\Theta) - M_r(\widehat{\Theta}) \Big| \end{aligned}$$

And if the first six moments of $F$ and $\widehat{F}$ match exactly, the right hand side is zero which is a contradiction. ■

So all we need to prove is that $F(x) - \widehat{F}(x)$ has at most six zero crossings. Let us prove a stronger lemma by induction:

**Lemma 6.5.4** *Let $f(x) = \sum_{i=1}^{k} \alpha_i \mathcal{N}(\mu_i, \sigma_i^2, x)$ be a linear combination of $k$ Gaussians ($\alpha_i$ can be negative). Then if $f(x)$ is not identically zero, $f(x)$ has at most $2k - 2$ zero crossings.*

We will rely on the following tools:

**Theorem 6.5.5** *Given $f(x) : \mathbb{R} \rightarrow \mathbb{R}$, that is analytic and has $n$ zero crossings, then for any $\sigma^2 > 0$, the function $g(x) = f(x) * \mathcal{N}(0, \sigma^2)$ has at most $n$ zero crossings.*

This theorem has a physical interpretation. If we think of $f(x)$ as the heat profile of an infinite one-dimensional rod, then what does the heat profile look like at some later time? In fact it is precisely $g(x) = f(x) * \mathcal{N}(0, \sigma^2)$ for an appropriately chosen $\sigma^2$. Alternatively, the Gaussian is the *Green's function* of the heat equation. And hence many of our physical intuitions for diffusion have consequences for convolution – convolving a function by a Gaussian has the effect of smoothing it, and it cannot create a new local maxima (and relatedly it cannot create new zero crossings).

Finally we recall the elementary fact:

**Fact 6.5.6** $\mathcal{N}(0, \sigma_1^2) * \mathcal{N}(0, \sigma_2^2) = \mathcal{N}(0, \sigma_1^2 + \sigma_2^2)$

Now we are ready to prove the above lemma and conclude that if we knew the first six moments of a mixture of two Gaussians *exactly*, then we would know its parameters exactly too. Let us prove the above lemma by induction, and assume that for any linear combination of $k = 3$ Gaussians, the number of zero crossings is

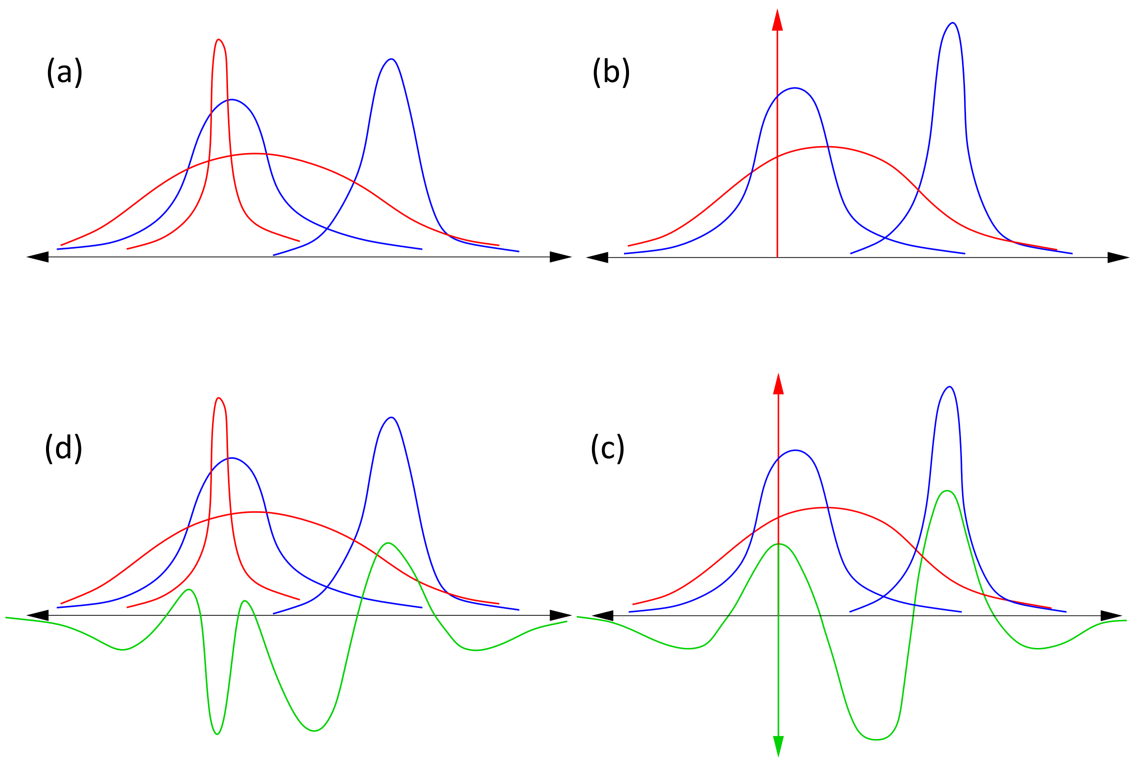

Figure 6.3: (a) linear combination of four Gaussians (b) subtracting $\sigma^2$ from each variance (c) adding back in the delta function (d) convolving by $\mathcal{N}(0, \sigma^2)$ to recover the original linear combination

at most four. Now consider an arbitrary linear combination of four Gaussians, and let $\sigma^2$ be the smallest variance of any component. See Figure 6.3(a). We can consider a related mixture where we subtract $\sigma^2$ from the variance of each component. See Figure 6.3(b).

Now if we ignore the delta function, we have a linear combination of three Gaussians and by induction we know that it has at most four zero crossings. But how many zero crossings can we add when we add back in the delta function? We can add at most two, one on the way up and one on the way down (here we are ignoring some real analysis complications of working with delta functions for ease of presentation). See Figure 6.3(c). And now we can convolve the function by $\mathcal{N}(0, \sigma^2)$ to recover the original linear combination of four Gaussians, but this last step does not increase the number of zero crossings! See Figure 6.3(d).

This proves that

$$\left\{ M_r(\widehat{\Theta}) = M_r(\Theta) \right\}, \ r = 1, 2, ..., 6$$

has only two solutions (the true parameters and we can also interchange which is component is which). In fact, this system of polynomial equations is also *stable* and there is an analogue of condition numbers for systems of polynomial equations that implies a quantitative version of what we have just proved: if $F$ and $\widehat{F}$ that are not $\varepsilon$-close, then one of their first six moments is noticeably different. This gives us our univariate algorithm.

## 6.6 A View from Algebraic Geometry

Here we will present an alternative univariate learning algorithm of Belkin and Sinha [23] that also makes use of the method of moments, but gives a much more general analysis using tools from algebraic geometry.

### Polynomial Families

We will analyze the method of moments for the following class of distributions:

**Definition 6.6.1** *A class of distributions $F(\Theta)$ is called a polynomial family if*

$$\forall r, \ \mathbb{E}_{X \in F(\Theta)}\left[X^r\right] = M_r(\Theta)$$

*where $M_r(\Theta)$ is a polynomial in $\Theta = (\theta_1, \theta_2, ...., \theta_k)$.*

This definition captures a broad class of distributions such as mixtures models whose components are uniform, exponential, Poisson, Gaussian or gamma functions. We will need another (tame) condition on the distribution which guarantees that it is characterized by all of its moments.

**Fact 6.6.2** *If the moment generating function (mgf) of $X$ defined as $\sum \mathbb{E}[X^n] \frac{t^n}{n!}$ converges in a neighborhood of zero, it uniquely determines the probability distribution, i.e.*

$$\forall r,\ M_r(\Theta) = M_r(\widehat{\Theta}) \implies F(\Theta) = F(\widehat{\Theta}).$$

Our goal is to show that for any polynomial family, a *finite* number of its moments suffice. First we introduce the relevant definitions:

**Definition 6.6.3** *Given a ring $R$, an ideal $I$ generated by $g_1, g_2, \cdots, g_n \in R$ denoted by $I = \langle g_1, g_2, \cdots, g_n \rangle$ is defined as*

$$I = \left\{ \sum_i r_i g_i \text{ where } r_i \in R \right\}.$$

**Definition 6.6.4** *A Noetherian ring is a ring such that for any sequence of ideals*

$$I_1 \subseteq I_2 \subseteq I_3 \subseteq \cdots,$$

*there is $N$ such that $I_N = I_{N+1} = I_{N+2} = \cdots$.*

**Theorem 6.6.5 (Hilbert's Basis Theorem)** *If $R$ is a Noetherian ring, then $R[X]$ is also a Noetherian ring.*

It is easy to see that $\mathbb{R}$ is a Noetherian ring, and hence we know that $\mathbb{R}[x]$ is also Noetherian. Now we can prove that for any polynomial family, a finite number of moments suffice to uniquely identify any distribution in the family:

**Theorem 6.6.6** *Let $F(\Theta)$ be a polynomial family. If the moment generating function converges in a neighborhood of zero, there exists $N$ such that*

$$F(\Theta) = F(\widehat{\Theta}) \text{ if and only if } M_r(\Theta) = M_r(\widehat{\Theta})\ \forall r \in 1, 2, \cdots, N$$

**Proof:** Let $Q_r(\Theta, \widehat{\Theta}) = M_r(\Theta) - M_r(\widehat{\Theta})$. Let $I_1 = \langle Q_1 \rangle,\ I_2 = \langle Q_1, Q_2 \rangle, \cdots$. This is our ascending chain of ideals in $\mathbb{R}[\Theta, \widehat{\Theta}]$. We can invoke Hilbert's basis theorem and conclude that $\mathbb{R}[X]$ is a Noetherian ring and hence, there is $N$ such that $I_N = I_{N+1} = \cdots$. So for all $N + j$, we have

$$Q_{N+j}(\Theta, \widehat{\Theta}) = \sum_{i=1}^{N} p_{ij}(\Theta, \widehat{\Theta}) Q_i(\Theta, \widehat{\Theta})$$

for some polynomial $p_{ij} \in \mathbb{R}[\Theta, \widehat{\Theta}]$. Thus, if $M_r(\Theta) = M_r(\widehat{\Theta})$ for all $r \in 1, 2, \cdots, N$, then $M_r(\Theta) = M_r(\widehat{\Theta})$ for all $r$ and from Fact 6.6.2 we conclude that $F(\Theta) = F(\widehat{\Theta})$.

The other side of the theorem is obvious. ■

The theorem above does not give any finite bound on $N$, since the basis theorem does not either. This is because the basis theorem is proved by contradiction, but more fundamentally it is not possible to give a bound on $N$ that depends only on the choice of the ring. Consider the following example

**Example 1** *Consider the Noetherian ring $\mathbb{R}[x]$. Let $I_i = \langle x^{N-i} \rangle$ for $i = 0, \cdots, N$. It is a strictly ascending chain of ideals for $i = 0, \cdots, N$. Therefore, even if the ring $\mathbb{R}[x]$ is fixed, there is no universal bound on $N$.*

Bounds such as those in Theorem 6.6.6 are often referred to as *ineffective*. Consider an application of the above result to mixtures of Gaussians: from the above theorem, we have that any two mixtures $F$ and $\widehat{F}$ of $k$ Gaussians are identical if and only if these mixtures agree on their first $N$ moments. Here $N$ is a function of $k$, and $N$ is finite but we cannot write down any explicit bound on $N$ as a function of $k$ using the above tools. Nevertheless, these tools apply much more broadly than the specialized ones based on the heat equation that we used to prove that $4k - 2$ moments suffice for mixtures of $k$ Gaussians in the previous section.

## Systems of Polynomial Inequalities

In general, we do not have exact access to the moments of a distribution but only noisy approximations. Our main goal is to prove a quantitive version of the previous result which shows that any two distributions $F$ and $\widehat{F}$ that are close on their first $N$ moments are close in their parameters too. The key fact is that we can bound the condition number of systems of polynomial inequalities; there are a number of ways to do this but we will use *quantifier elimination*. Recall:

**Definition 6.6.7** *A set $S$ is semi-algebraic if there exist multivariate polynomials $p_1, ..., p_n$ such that*

$$S = \{x_1, ..., x_r | p_i(x_1, ..., x_r) \geq 0\}$$

*or if $S$ is a finite union or intersection of such sets.*

**Theorem 6.6.8 (Tarski)** *The projection of a semi-algebraic set is semi-algebraic.*

We define the following helper set:

$$H(\varepsilon,\delta) = \left\{ \forall(\Theta, \widehat{\Theta}) \ : \ |M_r(\Theta) - M_r(\widehat{\Theta})| \leq \delta \text{ for } r = 1, 2, ...N \Longrightarrow \|\Theta - \widehat{\Theta}\| \leq \varepsilon \right\}.$$

Let $\varepsilon(\delta)$ be the smallest $\varepsilon$ as a function of $\delta$:

**Theorem 6.6.9** *There are fixed constants $C_1, C_2, s$ such that if $\delta < 1/C_1$ then $\varepsilon(\delta) < C_2\delta^{1/s}$.*

**Proof:** It is easy to see that we can define $H(\varepsilon,\delta)$ as the projection of a semi-algebraic set, and hence using Tarski's theorem we conclude that $H(\varepsilon,\delta)$ is also semi-algebraic. The crucial observation is that because $H(\varepsilon,\delta)$ is semi-algebraic, the smallest that we can choose $\varepsilon$ to be as a function of $\delta$ is itself a polynomial function of $\delta$. There are some caveats here, because we need to prove that for a fixed $\delta$ we can choose $\varepsilon$ to be strictly greater than zero and moreover the polynomial relationship between $\varepsilon$ and $\delta$ only holds if $\delta$ is sufficiently small. However these technical issues can be resolved without much more work, see [23] and the main result is the following. ■

**Corollary 6.6.10** *If $|M_r(\Theta) - M_r(\widehat{\Theta})| \leq \left(\frac{\varepsilon}{C_2}\right)^s$ then $|\Theta - \widehat{\Theta}| \leq \varepsilon$.*

Hence there is a polynomial time algorithm to learn the parameters of any univariate polynomial family (whose mgf converges in a neighborhood of zero) within an additive accuracy of $\varepsilon$ whose running time and sample complexity is poly$(1/\varepsilon)$; we can take enough samples to estimate the first $N$ moments within $\varepsilon^s$ and search over a grid of the parameters, and any set of parameters that matches each of the moments is necessarily close in parameter distance to the true parameters.

# Chapter 7

# Matrix Completion

Here we will give algorithms for the matrix completion problem, where we observe uniformly random entries of a low-rank, incoherent matrix $M$ and we would like design efficient algorithms that exactly recover $M$.

## 7.1 Background

The usual motivation for studying the matrix completion problem comes from recommendation systems. To be concrete, consider the Netflix problem where we are given ratings $M_{i,j}$ that represent how user $i$ rated movie $j$. We would like to use these ratings to make good recommendations to users, and a standard approach is to try to use our knowledge of some of the entries of $M$ to fill in the rest of $M$.

Let us be more precise: There is an unknown matrix $M \in \mathbb{R}^{n \times m}$ whose rows represent users and whose columns represent movies in the example above. For each $(i,j) \in \Omega \subseteq [n] \times [m]$ we are given the value $M_{i,j}$. Our goal is to recover $M$ exactly. Ideally, we would like to find the minimum rank matrix $X$ that agrees with $M$ on the observed entries $\{M_{i,j}\}_{(i,j)\in\Omega}$ however this problem is $NP$-hard. There are some now standard assumptions under which we will be able to give efficient algorithms for recovering $M$ exactly:

(a) $\Omega$ is uniformly random

(b) The singular vectors of $M$ are uncorrelated with the standard basis (such a matrix is called *incoherent* and we define this later)

In fact, we will see that there are efficient algorithms for recovering $M$ exactly if $m \approx mr \log m$ where $m \geq n$ and $\text{rank}(M) \leq r$. This is similar to compressed sensing, where we were able to recover a $k$-sparse signal $x$ from $O(k \log n/k)$ linear measurements, which is much smaller than the dimension of $x$. Here too we can recover a low-rank matrix $M$ from a number of observations that is much smaller than the dimension of $M$.

Let us examine the assumptions above. The assumption that should give us pause is that $\Omega$ is uniformly random. This is somewhat unnatural since it would be more believable if the probability we observe $M_{i,j}$ depends on the value itself. Alternatively, a user should be more likely to rate a movie *if he actually liked it.*

In order to understand the second assumption, suppose $\Omega$ is indeed uniformly random. Consider

$$M = \Pi \left[ \begin{array}{c|c} I_r & 0 \\ \hline 0 & 0 \end{array} \right] \Pi^T$$

where $\Pi$ is a uniformly random permutation matrix. $M$ is low-rank, but unless we observe all of the ones along the diagonal, we will not be able to recover $M$ uniquely. Indeed, the singular vectors of $M$ contain some of the standard basis vectors; but if we were to assume that the singular vectors of $M$ are incoherent with respect to the standard basis, we could avoid the above problem.

**Definition 7.1.1** *The coherence $\mu$ of a subspace $U \subseteq \mathbb{R}^n$ of dimension $dim(u) = r$ is*

$$\frac{n}{r} \max_i \|P_U e_i\|^2,$$

*where $P_U$ denotes the orthogonal projection onto $U$, and $e_i$ is the standard basis element.*

It is easy to see that if we choose $U$ uniformly at random, then $\mu(U) = \widetilde{O}(1)$. Also we have that $1 \leq \mu(U) \leq n/r$ and the upper bound is attained if $U$ contains any $e_i$. We can now see that if we set $U$ to be the top singular vectors of the above example, then $U$ has high coherence. We will need the following conditions on $M$:

(a) Let $M = U\Sigma V^T$, then $\mu(U), \mu(V) \leq \mu_0$.

(b) $\|UV^T\|_\infty \leq \frac{\mu_1 \sqrt{r}}{n}$, where $\|\cdot\|_\infty$ denotes the maximum absolute value of any entry.

The main result of this chapter is:

**Theorem 7.1.2** *Suppose $\Omega$ is chosen uniformly at random. Then there is a polynomial time algorithm to recover $M$ exactly that succeeds with high probability if*

$$m \geq \max(\mu_1^2, \mu_0) r(n+m) \log^2(n+m)$$

The algorithm in the theorem above is based on a convex relaxation for the rank of a matrix called the *nuclear norm*. We will introduce this in the next section, and establish some of its properties but one can think of it as an analogue to the $\ell_1$ minimization approach that we used in compressed sensing. This approach was first introduced in Fazel's thesis [58], and Recht, Fazel and Parrilo [104] proved that this approach exactly recovers $M$ in the setting of *matrix sensing*, which is related to the problem we consider here.

In a landmark paper, Candes and Recht [33] proved that the relaxation based on nuclear norm also succeeds for matrix completion and introduced the assumptions above in order to prove that their algorithm works. There has since been a long line of work improving the requirements on $m$, and the theorem above and our exposition will follow a recent paper of Recht [103] that greatly simplifies the analysis by making use of matrix analogues of the Bernstein bound and using these in a procedure now called *quantum golfing* that was first introduced by Gross [67].

**Remark 7.1.3** *We will restrict to $M \in \mathbb{R}^{n\times n}$ and assume $\mu_0, \mu_1 = \widetilde{O}(1)$ in our analysis, which will reduce the number of parameters we need to keep track of. Also let $m = n$.*

## 7.2 Nuclear Norm

Here we introduce the nuclear norm, which will be the basis for our algorithms for matrix completion. We will follow a parallel outline to that of compressed sensing. In particular, a natural starting point is the optimization problem:

$$(P0) \qquad \min \operatorname{rank}(X) \text{ s.t. } X_{i,j} = M_{i,j} \text{ for all } (i,j) \in \Omega$$

This optimization problem is $NP$-hard. If $\sigma(X)$ is the vector of singular values of $X$ then we can think of the rank of $X$ equivalently as the sparsity of $\sigma(X)$. Recall, in compressed sensing we faced a similar obstacle: finding the sparsest solution to a system of linear equations is also $NP$-hard, but we instead considered the $\ell_1$ relaxation and proved that under various conditions this optimization problem recovers the sparsest solution. Similarly it is natural to consider the $\ell_1$-norm of $\sigma(X)$ which is called the nuclear norm:

**Definition 7.2.1** *The nuclear norm of $X$ denoted by $\|X\|_*$ is $\|\sigma(X)\|_1$.*

We will instead solve the convex program:

$$(P1) \qquad \min \|X\|_* \text{ s.t. } X_{i,j} = M_{i,j} \text{ for all } (i,j) \in \Omega$$

and our goal is to prove conditions under which the solution to $(P1)$ is exactly $M$. Note that this is a convex program because $\|X\|_*$ is a norm, and there are a variety of efficient algorithms to solve the above program.

In fact, for our purposes a crucial notion is that of a *dual norm*. We will not need this concept in full-generality, so we state it for the specific case of the nuclear norm. This concept gives us a method to lower bound the nuclear norm of a matrix:

**Definition 7.2.2** *Let $\langle X, B\rangle = \sum_{i,j} X_{i,j}B_{i,j} = trace(X^T B)$ denote the matrix inner-product.*

**Lemma 7.2.3** $\|X\|_* = \max_{\|B\|\leq 1}\langle X, B\rangle$.

To get a feel for this, consider the special case where we restrict $X$ and $B$ to be diagonal. Moreover let $X = \text{diag}(x)$ and $B = \text{diag}(b)$. Then $\|X\|_* = \|x\|_1$ and the constraint $\|B\| \leq 1$ (the spectral norm of $B$ is at most one) is equivalent to $\|b\|_\infty \leq 1$. So we can recover a more familiar characterization of vector norms in the special case of diagonal matrices:

$$\|x\|_1 = \max_{\|b\|_\infty\leq 1} b^T x$$

**Proof:** We will only prove one direction of the above lemma. What $B$ should we use to certify the nuclear norm of $X$. Let $X = U_X\Sigma_X V_X^T$, then we will choose $B = U_X V_X^T$. Then

$$\langle X, B\rangle = \text{trace}(B^T X) = \text{trace}(V_X U_X^T U_X \Sigma_X V_X^T) = \text{trace}(V_X\Sigma_X V_X^T) = \text{trace}(\Sigma_X) = \|X\|_*$$

where we have used the basic fact that $\text{trace}(ABC) = \text{trace}(BCA)$. Hence this proves $\|X\|_* \leq \max_{\|B\|\leq 1}\langle X, B\rangle$, and the other direction is not much more difficult (see e.g. [74]). ■

How can we show that the solution to $(P1)$ is $M$? Our basic approach will be a proof by contradiction. Suppose not, then the solution is $M+Z$ for some $Z$ that is supported in $\overline{\Omega}$. Our goal will be to construct a matrix $B$ of spectral norm at most one for which

$$\|M+Z\|_* \geq \langle M+Z, B\rangle > \|M\|_*$$

Hence $M+Z$ would not be the optimal solution to $(P1)$. This strategy is similar to the one in compressed sensing, where we hypothesized some other solution $w$ that differs from $x$ by a vector $y$ in the kernel of the sensing matrix $A$. We used geometric properties of $\ker(A)$ to prove that $w$ has strictly larger $\ell_1$ norm than $x$. However the proof here will be more involved since our strategy is to construct $B$ above based on $Z$ (rather than relying on some geometry property of $A$ that holds regardless of what $y$ is).

Let us introduce some basic projection operators that will be crucial in our proof. Recall, $M = U\Sigma V^T$, let $u_1, \dots, u_r$ be columns of $U$ and let $v_1, \dots, v_r$ be columns of $V$. Choose $u_{r+1}, \dots, u_n$ so that $u_1, \dots, u_n$ form an orthonormal basis for all of $\mathbb{R}^n$ – i.e. $u_{r+1}, \dots, u_n$ is an arbitrary orthonormal basis of $U^\perp$. Similarly choose $v_{r+1}, \dots, v_n$ so that $v_1, \dots, v_n$ form an orthonormal basis for all of $\mathbb{R}^n$. We will be interested in the following linear spaces over matrices:

**Definition 7.2.4** $T = \mathit{span}\{u_i v_j^T \mid 1 \le i \le r \ \mathit{or}\ 1 \le j \le r \ \mathit{or\ both}\}$.

Then $T^\perp = \text{span}\{u_i v_j^T \text{ s.t. } r+1 \le i, j \le n\}$.. We have $\dim(T) = r^2 + 2(n-r)r$ and $\dim(T^\perp) = (n-r)^2$. Moreover we can define the linear operators that project into $T$ and $T^\perp$ respectively:

$$P_{T^\perp}[Z] = \sum_{i,j=r+1}^{n} \langle Z, u_i v_j^T \rangle \cdot U_i v_j^T = P_{U^\perp} Z P_{V^\perp}.$$

And similarly

$$P_T[Z] = \sum_{(i,j)\in[n]\times[n]-[r+1,n]\times[r+1,n]} \langle Z, u_i v_j^T \rangle \cdot u_i v_j^T = P_U Z + Z P_V - P_U Z P_V.$$

We are now ready to describe the outline of the proof of Theorem 7.1.2. The proof will be based on:

(a) We will assume that a certain helper matrix $Y$ exists, and show that this is enough to imply $\|M+Z\|_* > \|M\|_*$ for any $Z$ supported in $\Omega$

(b) We will construct such a $Y$ using quantum golfing [67].

## Part $(a)$

Here we will state the conditions we need on the helper matrix $Y$ and prove that if such a $Y$ exists, then $M$ is the solution to $(P1)$. We require that $Y$ is supported in $\Omega$ and

(a) $\|P_T(Y) - UV^T\|_F \le \sqrt{r/8n}$

(b) $\|P_{T^\perp}(Y)\| \leq 1/2$.

We want to prove that for any $Z$ supported in $\overline{\Omega}$, $\|M+Z\|_* > \|M\|_*$. Recall, we want to find a matrix $B$ of spectral norm at most one so that $\langle M+Z, B\rangle > \|M\|_*$. Let $U_\perp$ and $V_\perp$ be singular vectors of $P_{T^\perp}[Z]$. Then consider

$$B = \begin{bmatrix} U & U_\perp \end{bmatrix} \cdot \begin{bmatrix} V^T \\ V_\perp^T \end{bmatrix} = UV^T + U_\perp V_\perp^T.$$

**Claim 7.2.5** $\|B\| \leq 1$

**Proof:** By construction $U^T U_\perp = 0$ and $V^T V_\perp = 0$ and hence the above expression for $B$ is its singular value decomposition, and the claim now follows. ■

Hence we can plug in our choice for $B$ and simplify:

$$\begin{aligned} \|M+Z\|_* &\geq \langle M+Z, B\rangle \\ &= \langle M+Z, UV^T + U_\perp V_\perp^T\rangle \\ &= \underbrace{\langle M, UV^T\rangle}_{\|M\|_*} + \langle Z, UV^T + U_\perp V_\perp^T\rangle \end{aligned}$$

where in the last line we used the fact that $M$ is orthogonal to $U_\perp V_\perp^T$. Now using the fact that $Y$ and $Z$ have disjoint supports we can conclude:

$$\|M+Z\|_* \geq \|M\|_* + \langle Z, UV^T + U_\perp V_\perp^T - Y\rangle$$

Therefore in order to prove the main result in this section it suffices to prove that $\langle Z, UV^T + U_\perp V_\perp^T - Y\rangle > 0$. We can expand this quantity in terms of its projection onto $T$ and $T^\perp$ and simplify as follows:

$$\begin{aligned} \|M+Z\|_* - \|M\|_* &\geq \langle P_T(Z), P_T(UV^T + U_\perp V_\perp^T - Y)\rangle + \langle P_{T^\perp}(Z), P_{T^\perp}(UV^T + U_\perp V_\perp^T - Y)\rangle \\ &\geq \langle P_T(Z), UV^T - P_T(Y)\rangle + \langle P_{T^\perp}(Z), U_\perp V_\perp^T - P_{T^\perp}(Y)\rangle \\ &\geq \langle P_T(Z), UV^T - P_T(Y)\rangle + \|P_{T^\perp}(Z)\|_* - \langle P_{T^\perp}(Z), P_{T^\perp}(Y)\rangle \end{aligned}$$

where in the last line we used the fact that $U_\perp$ and $V_\perp$ are the singular vectors of $P_{T^\perp}[Z]$ and hence $\langle U_\perp V_\perp^T, P_{T^\perp}[Z]\rangle = \|P_{T^\perp}[Z]\|_*$.

Now we can invoke the properties of $Y$ that we have assumed in this section, to prove a lower bound on the right hand side. By property $(a)$ of $Y$, we have that $\|P_T(Y) - UV^T\|_F \leq \sqrt{\frac{r}{2n}}$. Therefore, we know that the first term $\langle P_T(Z), UV^T - P_T(Y)\rangle \geq -\sqrt{\frac{r}{8n}}\|P_T(Z)\|_F$. By property $(b)$ of $Y$, we know the operator norm of $P_T^\perp(Y)$ is at most $1/2$. Therefore the third term $\langle P_{T^\perp}(Z), P_{T^\perp}(Y)\rangle$ is at most $\frac{1}{2}\|P_{T^\perp}(Z)\|_*$. Hence

$$\|M+Z\|_* - \|M\|_* \geq -\sqrt{\frac{r}{8n}}\|P_T(Z)\|_F + \frac{1}{2}\|P_{T^\perp}(Z)\|_* \overset{?}{>} 0$$

We will show that with high probability over the choice of $\Omega$ that the inequality does indeed hold. We defer the proof of this last fact, since it and the construction of the helper matrix $Y$ will both make use of the matrix Bernstein inequality which we present in the next section.

## 7.3 Quantum Golfing

What remains is to construct a helper matrix $Y$ and prove that with high probability over $\Omega$, for any matrix $Z$ supported in $\overline{\Omega}$ that $\|P_{T^\perp}(Z)\|_* > \sqrt{\frac{r}{2n}}\|P_T(Z)\|_F$ to complete the proof we started in the previous section. We will make use of an approach introduced by Gross [67] and we will follow the proof of Recht in [103] where the strategy is to construct $Y$ iteratively. In each phase, we will invoke concentration results for matrix valued random variables to prove that the error part of $Y$ decreases geometrically and we make rapid progress in constructing a good helper matrix.

First we will introduce the key concentration result that we will apply in several settings. The following matrix valued Bernstein inequality first appeared in the work of Ahlswede and Winter related to quantum information theory [6].

**Theorem 7.3.1 (Non-commutative Bernstein Inequality)** *Let $X_1 \dots X_l$ be independent mean 0 matrices of size $d \times d$. Let $\rho_k^2 = \max\{\|\mathbb{E}[X_k X_k^T]\|, \|\mathbb{E}[X_k^T X_k]\|\}$ and suppose $\|X_k\| \leq M$ almost surely. Then for $\tau > 0$,*

$$\Pr\left[\Big\|\sum_{k=1}^{l} X_k\Big\| > \tau\right] \leq 2d\exp\left\{\frac{-\tau^2/2}{\sum_k \rho_k^2 + M\tau/3}\right\}$$

If $d = 1$ this is the standard Bernstein inequality. If $d > 1$ and the matrices $X_k$ are diagonal then this inequality can be obtained from the union bound and the standard Bernstein inequality again. However to build intuition, consider the following toy problem. Let $u_k$ be a random unit vector in $\mathbb{R}^d$ and let $X_k = u_k u_k^T$. Then it is easy to see that $\rho_k^2 = 1/d$. How many trials do we need so that $\sum_k X_k$ is close to the identity (after scaling)? We should expect to need $\Theta(d \log d)$ trials; this is even true if $u_k$ is drawn uniformly at random from the standard basis vectors $\{e_1 \dots e_d\}$ due to the coupon collector problem. Indeed, the above bound corroborates our intuition that $\Theta(d \log d)$ is necessary and sufficient.

Now we will apply the above inequality to build up the tools we will need to finish the proof.

**Definition 7.3.2** *Let $R_\Omega$ be the operator that zeros out all the entries of a matrix except those in $\Omega$.*

**Lemma 7.3.3** *If $\Omega$ is chosen uniformly at random and $m \geq nr \log n$ then with high probability*

$$\frac{n^2}{m} \left\| P_T R_\Omega P_T - \frac{m}{n^2} P_T \right\| < \frac{1}{2}$$

**Remark 7.3.4** *Here we are interested in bounding the operator norm of a linear operator on matrices. Let $T$ be such an operator, then $\|T\|$ is defined as*

$$\max_{\|Z\|_F \leq 1} \|T(Z)\|_F$$

We will explain how this bound fits into the framework of the matrix Bernstein inequality, but for a full proof see [103]. Note that $\mathbb{E}[P_T R_\Omega P_T] = P_T \, \mathbb{E}[R_\Omega] P_T = \frac{m}{n^2} P_T$ and so we just need to show that $P_T R_\Omega P_T$ does not deviate too far from its expectation. Let $e_1, e_2, \ldots, e_d$ be the standard basis vectors. Then we can expand:

$$\begin{aligned} P_T(Z) &= \sum_{a,b} \langle P_T(Z), e_a e_b^T \rangle e_a e_b^T \\ &= \sum_{a,b} \langle P_T(Z), e_a e_b^T \rangle e_a e_b^T \\ &= \sum_{a,b} \langle Z, P_T(e_a e_b^T) \rangle e_a e_b^T \end{aligned}$$

Hence $R_\Omega P_T(Z) = \sum_{(a,b) \in \Omega} \left\langle Z, P_T(e_a e_b^T) \right\rangle e_a e_b^T$ and finally we conclude that

$$P_T R_\Omega P_T(Z) = \sum_{(a,b) \in \Omega} \left\langle Z, P_T(e_a e_b^T) \right\rangle P_T(e_a e_b^T)$$

We can think of $P_T R_\Omega P_T$ as the sum of random operators of the form $\tau_{a,b} : Z \to \left\langle Z, P_T(e_a e_b^T) \right\rangle P_T(e_a e_b^T)$, and the lemma follows by applying the matrix Bernstein inequality to the random operator $\sum_{(a,b) \in \Omega} \tau_{a,b}$.

We can now complete the deferred proof of part $(a)$:

**Lemma 7.3.5** *If $\Omega$ is chosen uniformly at random and $m \geq nr \log n$ then with high probability for any $Z$ supported in $\overline{\Omega}$ we have*

$$\|P_{T^\perp}(Z)\|_* > \sqrt{\frac{r}{2n}}\|P_T(Z)\|_F$$

**Proof:** Using Lemma 7.3.3 and the definition of the operator norm (see the remark) we have

$$\left\langle Z, P_T R_\Omega P_T Z - \frac{m}{n^2} P_T Z \right\rangle \geq -\frac{m}{2n^2}\|Z\|_F^2$$

Furthermore we can upper bound the left hand side as:

$$\begin{aligned}\langle Z, P_T R_\Omega P_T Z\rangle = \langle Z, P_T R_\Omega^2 P_T Z\rangle &= \|R_\Omega(Z - P_{T^\perp}(Z))\|_F^2 \\ &= \|R_\Omega(P_{T^\perp}(Z))\|_F^2 \leq \|P_{T^\perp}(Z)\|_F^2\end{aligned}$$

where in the last line we used that $Z$ is supported in $\overline{\Omega}$ and so $R_\Omega(Z) = 0$. Hence we have that

$$\|P_{T^\perp}(Z)\|_F^2 \geq \frac{m}{n^2}\|P_T(Z)\|_F^2 - \frac{m}{2n^2}\|Z\|_F^2$$

We can use the fact that $\|Z\|_F^2 = \|P_{T^\perp}(Z)\|_F^2 + \|P_T(Z)\|_F^2$ and conclude $\|P_{T^\perp}(Z)\|_F^2 \geq \frac{m}{4n^2}\|P_T(Z)\|_F^2$. We can now complete the proof of the lemma

$$\begin{aligned}\|P_{T^\perp}(Z)\|_*^2 \geq \|P_{T^\perp}(Z)\|_F^2 &\geq \frac{m}{4n^2}\|P_T(Z)\|_F^2 \\ &> \frac{r}{2n}\|P_T(Z)\|_F^2\end{aligned}$$

■

All that remains is to prove that the helper matrix $Y$ that we made use of actually does exists (with high probability). Recall that we require that $Y$ is supported in $\Omega$ and $\|P_T(Y) - UV^T\|_F \leq \sqrt{r/8n}$ and $\|P_{T^\perp}(Y)\| \leq 1/2$. The basic idea is to break up $\Omega$ into disjoint sets $\Omega_1, \Omega_2, \dots \Omega_p$, where $p = \log n$ and use each set of observations to make progress on the remained $P_T(Y) - UV^T$. More precisely, initialize $Y_0 = 0$ in which case the remainder is $W_0 = UV^T$. Then set

$$Y_{i+1} = Y_i + \frac{n^2}{m} R_{\Omega_{i+1}}(W_i)$$

and update $W_{i+1} = UV^T - P_T(Y_{i+1})$. It is easy to see that $\mathbb{E}[\frac{n^2}{m} R_{\Omega_{i+1}}] = I$. Intuitively this means that at each step $Y_{i+1} - Y_i$ is an unbiased estimator for $W_i$ and so we should expect the remainder to decrease quickly (here we will rely on the concentration bounds we derived from the non-commutative Bernstein inequality). Now we can explain the nomenclature *quantum golfing*; at each step, we hit our golf ball in the direction of the hole but here our target is to approximate the matrix $UV^T$ which for various reasons is the type of question that arises in quantum mechanics.

It is easy to see that $Y = \sum_i Y_i$ is supported in $\Omega$ and that $P_T(W_i) = W_i$ for all $i$. Hence we can compute

$$\begin{aligned}\|P_T(Y_i) - UV^T\|_F = \left\|W_{i-1} - P_T\frac{n^2}{m}R_{\Omega_i}W_{i-1}\right\|_F &= \left\|P_TW_{i-1} - P_T\frac{n^2}{m}R_{\Omega_i}P_TW_{i-1}\right\|_F \\ \leq \frac{n^2}{m}\left\|P_TR_\Omega P_T - \frac{m}{n^2}P_T\right\| &\leq \frac{1}{2}\|W_{i-1}\|_F\end{aligned}$$

where the last inequality follows from Lemma 7.3.3. Therefore the Frobenius norm of the remainder decreases geometrically and it is easy to guarantee that $Y$ satisfies condition $(a)$.

The more technically involved part is showing that $Y$ also satisfies condition $(b)$. However the intuition is that $\|P_{T^\perp}(Y_1)\|$ is itself not too large, and since the norm of the remainder $W_i$ decreases geometrically we should expect that $\|P_{T^\perp}(Y_i)\|$ does too and so most of the contribution to

$$\|P_{T^\perp}(Y)\| \leq \sum_i \|P_{T^\perp}(Y_i)\|$$

comes from the first term. For the full details see [103]. This completes the proof that computing the solution to the convex program indeed finds $M$ *exactly*, provided that $M$ is incoherent and $|\Omega| \geq \max(\mu_1^2, \mu_0)r(n+m)\log^2(n+m)$.

# Bibliography

[1] D. Achlioptas and F. McSherry. On spectral learning of mixtures of distributions. In *COLT*, pages 458–469, 2005.

[2] A. Agarwal, A. Anandkumar, P. Jain, P. Netrapalli, R. Tandon Learning sparsely used overcomplete dictionaries via alternating minimization *arxiv:1310.7991*, 2013

[3] A. Agarwal, A. Anandkumar, P. Netrapalli Exact recovery of sparsely used overcomplete dictionaries *arxiv:1309.1952*, 2013

[4] M. Aharon. *Overcomplete Dictionaries for Sparse Representation of Signals.* PhD Thesis, 2006.

[5] M. Aharon, M. Elad and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. on Signal Processing*, 54(11):4311–4322, 2006.

[6] R. Ahlswede and A. Winter. Strong converse for identification via quantum channels. *IEEE Trans. Inf. Theory* 48(3):569–579, 2002.

[7] Noga Alon. *Tools from Higher Algebra.* In *Handbook of Combinatorics*, pages 1749–1783, 1996.

[8] A. Anandkumar, D. Foster, D. Hsu, S. Kakade, Y. Liu. A spectral algorithm for latent dirichlet allocation. In *NIPS*, pages 926–934, 2012.

[9] A. Anandkumar, R. Ge, D. Hsu and S. Kakade. A tensor spectral approach to learning mixed membership community models. In *COLT*, pages 867–881, 2013.

[10] A. Anandkumar, D. Hsu and S. Kakade. A method of moments for hidden markov models and multi-view mixture models. In *COLT*, pages 33.1–33.34, 2012.

[11] J. Anderson, M. Belkin, N. Goyal, L Rademacher and J. Voss. The more the merrier: the blessing of dimensionality for learning large Gaussian mixtures. *arxiv:1311.2891*, 2013.

[12] S. Arora, R. Ge, Y. Halpern, D. Mimno, A. Moitra, D. Sontag, Y. Wu and M. Zhu. A practical algorithm for topic modeling with provable guarantees. In *ICML*, pages 280–288, 2013.

[13] S. Arora, R. Ge, R. Kannan and A. Moitra. Computing a nonnegative matrix factorization – provably In *STOC*, pages 145–162, 2012.

[14] S. Arora, R. Ge and A. Moitra. Learning topic models - going beyond SVD. In *FOCS*, pages 1–10, 2012.

[15] S. Arora, R. Ge and A. Moitra. New algorithms for learning incoherent and overcomplete dictionaries. *arxiv:1308.6273*, 2013

[16] S. Arora, R. Ge, A. Moitra and S. Sachdeva. Provable ICA with unknown gaussian noise, and implications for gaussian mixtures and autoencoders. In *NIPS*, pages 2384–2392, 2012.

[17] S. Arora, R. Ge, S. Sachdeva and G. Schoenebeck. Finding overlapping communities in social networks: Towards a rigorous approach. In *EC*, 2012.

[18] S. Arora and R. Kannan. Learning mixtures of separated nonspherical gaussians. *Annals of Applied Probability*, pages 69-92, 2005.

[19] M. Balcan, A. Blum and A. Gupta. Clustering under approximation stability. *Journal of the ACM*, 2013.

[20] M. Balcan, A. Blum and N. Srebro. On a theory of learning with similarity functions. *Machine Learning*, pages 89–112, 2008.

[21] M. Balcan, C. Borgs, M. Braverman, J. Chayes and S-H Teng. Finding endogenously formed communities. In *SODA*, 2013.

[22] M. Belkin and K. Sinha. Toward learning gaussian mixtures with arbitrary separation. In *COLT*, pages 407–419, 2010.

[23] M. Belkin and K. Sinha. Polynomial learning of distribution families. In *FOCS*, pages 103–112, 2010.

[24] Q. Berthet and P. Rigollet. Complexity theoretic lower bounds for sparse principal component detection. In *COLT*, pages 1046–1066, 2013.

[25] A. Bhaskara, M. Charikar and A. Vijayaraghavan. Uniqueness of tensor decompositions with applications to polynomial identifiability. *arxiv:1304.8087*, 2013.

[26] A. Bhaskara, M. Charikar, A. Moitra and A. Vijayaraghavan. Smoothed analysis of tensor decompositions. In *STOC*, 2014.

[27] V. Bittorf, B. Recht, C. Re, and J. Tropp. Factoring nonnegative matrices with linear programs. In *NIPS*, 2012.

[28] D. Blei. Introduction to probabilistic topic models. *Communications of the ACM*, pages 77–84, 2012.

[29] D. Blei and J. Lafferty. A correlated topic model of Science. *Annals of Applied Statistics*, pp. 17–35, 2007.

[30] D. Blei, A. Ng and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, pages 993–1022, 2003.

[31] A. Blum, A. Kalai and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM* 50: 506-519, 2003.

[32] S. C. Brubaker and S. Vempala. Isotropic PCA and affine-invariant clustering. In *FOCS*, pages 551–560, 2008.

[33] E. Candes and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Math.*, pages 717–772, 2008.

[34] E. Candes, J. Romberg and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications of Pure and Applied Math.*, pp. 1207–1223, 2006.

[35] E. Candes and T. Tao. Decoding by linear programming. *IEEE Trans. on Information Theory*, 51(12):4203–4215, 2005.

[36] J. Chang. Full reconstruction of markov models on evolutionary trees: identifiability and consistency. *Mathematical Biosciences*, 137(1):51–73, 1996.

[37] K. Chaudhuri and S. Rao. Learning mixtures of product distributions using correlations and independence. In *COLT*, pages 9–20, 2008.

[38] K. Chaudhuri and S. Rao. Beyond Gaussians: Spectral methods for learning mixtures of heavy-tailed distributions. In *COLT*, pages 21–32, 2008.

[39] S. Chen, D. Donoho and M. Saunders. Atomic decomposition by basis pursuit. *SIAM J. on Scientific Computing*, 20(1):33–61, 1998.

[40] A. Cohen, W. Dahmen and R. DeVore. Compressed sensing and best $k$-term approximation. *Journal of the AMS*, pages 211–231, 2009.

[41] J. Cohen and U. Rothblum. Nonnegative ranks, decompositions and factorizations of nonnegative matrices. *Linear Algebra and its Applications*, pages 149–168, 1993.

[42] P. Comon. Independent component analysis: A new concept? *Signal Processing*, pages 287–314, 1994.

[43] A. Dasgupta. *Asymptotic Theory of Statistics and Probability.* Springer, 2008.

[44] A. Dasgupta, J. Hopcroft, J. Kleinberg, and M. Sandler. On learning mixtures of heavy-tailed distributions. In *FOCS*, pages 491–500, 2005.

[45] S. Dasgupta. Learning mixtures of gaussians. In *FOCS*, pages 634–644, 1999.

[46] S. Dasgupta and L. J. Schulman. A two-round variant of EM for gaussian mixtures. In *UAI*, pages 152–159, 2000.

[47] G. Davis, S. Mallat and M. Avellaneda. Greedy adaptive approximations. *J. of Constructive Approximation*, 13:57–98, 1997.

[48] L. De Lathauwer, J Castaing and J. Cardoso. Fourth-order Cumulant-based Blind Identification of Underdetermined Mixtures. *IEEE Trans. on Signal Processing*, 55(6):2965–2973, 2007.

[49] S. Deerwester, S. Dumais, T. Landauer, G. Furnas and R. Harshman. Indexing by latent semantic analysis. *JASIS*, pages 391–407, 1990.

[50] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B*, pages 1–38, 1977.

[51] D. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via $\ell_1$-minimization. *PNAS*, 100(5):2197–2202, 2003.

[52] D. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Trans. on IT*, 47(7):2845–2862, 1999.

[53] D. Donoho and P. Stark. Uncertainty principles and signal recovery. *SIAM J. on Appl. Math.*, 49(3):906–931, 1989.

[54] D. Donoho and V. Stodden. When does nonnegative matrix factorization give the correct decomposition into parts? In *NIPS*, 2003.

[55] M. Elad. *Sparse and Redundant Representations*. Springer, 2010.

[56] K. Engan, S. Aase and J. Hakon-Husoy. Method of optimal directions for frame design. *ICASSP*, 5:2443–2446, 1999.

[57] P. Erdos, M. Steel, L. Szekely and T. Warnow. A few logs suffice to build (almost) all trees. I. *Random Structures and Algorithms* 14:153-184, 1997.

[58] M. Fazel. *Matrix Rank MInimization with Applications.* PhD thesis, Stanford University, 2002.

[59] U. Feige and R. Krauthgamer. Finding and certifying a large hidden clique in a semirandom graph. *Random Structures and Algorithms*, pages 195–208, 2009.

[60] U. Feige and J. Kilian. Heuristics for semi random graph problems. *JCSS*, pages 639–671, 2001.

[61] J. Feldman, R. A. Servedio, and R. O'Donnell. PAC learning axis-aligned mixtures of gaussians with no separation assumption. In *COLT*, pages 20–34, 2006.

[62] A. Frieze, M. Jerrum, R. Kannan. Learning linear transformations. In *FOCS*, pages 359–368, 1996.

[63] A. Garnaev and E. Gluskin. The widths of a Euclidean ball. *Sovieth Math. Dokl.*, pages 200–204, 1984.

[64] A. Gilbert, S. Muthukrishnan and M. Strauss. Approximation of functions over redundant dictionaries using coherence. In *SODA*, 2003.

[65] N. Gillis. Robustness analysis of hotttopixx, a linear programming model for factoring nonnegative matrices. *arxiv:1211.6687*, 2012.

[66] N. Goyal, S. Vempala and Y. Xiao. Fourier PCA. In *STOC*, 2014.

[67] D. Gross. Recovering low-rank matrices from few coefficients in any basis. *arxiv:0910.1879*, 2009.

[68] D. Gross, Y-K Liu, S. Flammia, S. Becker and J. Eisert. Quantum state tomography via compressed sensing. *Physical Review Letters*, 105(15), 2010.

[69] V. Guruswami, J. Lee, and A. Razborov. Almost euclidean subspaces of $\ell_1^n$ via expander codes. *Combinatorica*, 30(1):47–68, 2010.

[70] R. Harshman. Foundations of the PARFAC procedure: model and conditions for an 'explanatory' multi-mode factor analysis. *UCLA Working Papers in Phonetics*, pages 1–84, 1970.

[71] J. Håstad. Tensor rank is $NP$-complete. *Journal of Algorithms*, 11(4):644-654, 1990.

[72] C. Hillar and L-H. Lim. Most tensor problems are $NP$-hard. *arxiv:0911.1393v4*, 2013

[73] T. Hofmann. Probabilistic latent semantic analysis. In *UAI* , pages 289–296, 1999.

[74] R. Horn and C. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

[75] D. Hsu and S. Kakade. Learning mixtures of spherical gaussians: Moment methods and spectral decompositions. In *ITCS*, pages 11–20, 2013.

[76] P. J. Huber. Projection pursuit. *Annals of Statistics* 13:435–475, 1985.

[77] R. A. Hummel and B. C. Gidas. Zero crossings and the heat equation. *Courant Institute of Mathematical Sciences* TR-111, 1984.

[78] R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *J. Computer and System Sciences* 62(2):pp. 367–375, 2001.

[79] A. T. Kalai, A. Moitra, and G. Valiant. Efficiently learning mixtures of two gaussians. In *STOC*, pages 553-562, 2010.

[80] B. Kashin and V. Temlyakov. A remark on compressed sensing. Manuscript, 2007.

[81] L. Khachiyan. On the complexity of approximating extremal determinants in matrices. *Journal of Complexity*, pages 138–153, 1995.

[82] D. Koller and N. Friedman. *Probabilistic Graphical Models*. MIT Press, 2009.

[83] J. Kruskal. Three-way arrays: Rank and uniqueness of trilinear decompositions with applications to arithmetic complexity and statistics. *Linear Algebra and its Applications*, pages 95–138, 1997.

[84] A. Kumar, V. Sindhwani and P. Kambadur. Fast conical hull algorithms for near-separable non-negative matrix factorization. In *ICML*, pages 231–239, 2013.

[85] D. Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, pages 788-791, 1999.

[86] D. Lee and H. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

[87] S. Leurgans, R. Ross and R. Abel. A decomposition for three-way arrays. *SIAM Journal on Matrix Analysis and Applications*, 14(4):1064–1083, 1993.

[88] M. Lewicki and T. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12:337–365, 2000.

[89] W. Li and A. McCallum. Pachinko allocation: DAG-structured mixture models of topic correlations. *ICML*, pp. 633-640, 2007.

[90] B. Lindsay. *Mixture Models: Theory, Geometry and Applications.* Institute for Mathematical Statistics, 1995.

[91] F. McSherry. Spectral partitioning of random graphs. In *FOCS*, pages 529–537, 2001.

[92] S. Mallat. *A Wavelet Tour of Signal Processing.* Academic-Press, 1998.

[93] S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Trans. on Signal Processing*, 41(12):3397–3415, 1993.

[94] A. Moitra. An almost optimal algorithm for computing nonnegative rank. In *SODA*, pages 1454–1464, 2003.

[95] A. Moitra and G. Valiant. Setting the polynomial learnability of mixtures of gaussians. In *FOCS*, pages 93–102, 2010.

[96] E. Mossel and S. Roch. Learning nonsingular phylogenies and hidden markov models. In *STOC*, pages 366–375, 2005.

[97] B. Olshausen and B. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):331–3325, 1997.

[98] C. Papadimitriou, P. Raghavan, H. Tamaki and S. Vempala. Latent semantic indexing: A probabilistic analysis. *JCSS*, pages 217–235, 2000.

[99] Y. Pati, R. Rezaiifar, P. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. *Asilomar Conference on Signals, Systems and Computers*, pages 40–44, 1993.

[100] K. Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society A*, 1894.

[101] Y. Rabani, L. Schulman and C. Swamy. Learning mixtures of arbitrary distributions over large discrete domains. . In *ITCS* 2014.

[102] R. Raz. Tensor-rank and lower bounds for arithmetic formulas. In *STOC*, pages 659–666, 2010.

[103] B. Recht. A simpler approach to matrix completion. *Journal of Machine Learning Research*, pages 3413–3430, 2011.

[104] B. Recht, M. Fazel and P. Parrilo. Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization. *SIAM Review*, pages 471–501, 2010.

[105] R. A. Redner and H. F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195-239, 1984.

[106] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. Journal of Symbolic Computation, pages 255-352, 1991.

[107] A. Seidenberg. A new decision method for elementary algebra. *Annals of Math*, pages 365–374, 1954.

[108] D. Spielman, H. Wang and J. Wright. Exact recovery of sparsely-used dictionaries. *Journal of Machine Learning Research*, 2012.

[109] M. Steel. Recovering a tree from the leaf colourations it generates under a Markov model. *Appl. Math. Lett.* 7:19-24, 1994.

[110] A. Tarski. A decision method for elementary algebra and geometry. *University of California Press*, 1951.

[111] H. Teicher. Identifiability of mixtures. *Annals of Mathematical Statistics*, pages 244–248, 1961.

[112] J. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. on IT*, 50(10):2231–2242, 2004.

[113] J. Tropp, A. Gilbert, S. Muthukrishnan and M. Strauss. Improved sparse approximation over quasi-incoherent dictionaries. *IEEE International Conf. on Image Processing*, 2003.

[114] L. Valiant. A theory of the learnable. *Comm. ACM*, 27(11):1134–1142, 1984.

[115] S. Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, pages 1364-1377, 2009.

[116] S. Vempala, Y. Xiao. Structure from local optima: Learning subspace juntas via higher order PCA. *Arxiv:*abs/1108.3329, 2011.

[117] S. Vempala and G. Wang. A spectral algorithm for learning mixture ,odels. *Journal of Computer and System Sciences*, pages 841–860, 2004.

[118] M. Wainwright and M. Jordan. *Graphical Models, Exponential Families, and Variational Inference. Foundations and Trends in Machine Learning*, pages 1–305, 2008.

[119] P. Wedin. Perturbation bounds in connection with singular value decompositions. *BIT*, 12:99–111, 1972.